2013-1251, -1252

United States Court of Appeals for the Federal Circuit

DATATERN, INC.,

Plaintiff-Appellant,

v.

BLAZENT, INC.,

Defendant-Appellee,

and

EPICOR SOFTWARE CORPORATION,

Defendant-Appellee,

and

INFORMATICA CORPORATION,

Defendant-Appellee,

(Caption Continued Inside)

Appeal from the United States District Court for the District of Massachusetts in consolidated Nos. 11-CV-11970 and 11-CV-12220, Judge F. Dennis Saylor, IV

BRIEF OF PLAINTIFF-APPELLANT

LEE CARL BROMBERG
ERIK PAUL BELT
MCCARTER & ENGLISH, LLP
265 Franklin Street
Boston, MA 02110
(617) 449-6506

Attorneys for Plaintiff-Appellant

MAY 28, 2013

CAPTION CONTINUED

and

CARL WARREN & CO., INC.,

Defendant-Appellee,

and

LANCET SOFTWARE DEVELOPMENT, INC.,

Defendant-Appellee,

and

MAGIC SOFTWARE ENTERTAINMENT, INC.,
and MAGIC SOFTWARE ENTERTAINMENT, LTD,

Defendants-Appellees,

and

TERADATA CORPORATION,

Defendant-Appellee,

and

PREMIER, INC.,

Defendant-Appellee,

and

MICROSTRATEGY, INC.,

Defendant-Appellee,

and

AIRLINES REPORTING CORPORATION,

Defendant-Appellee.

Form 9

FORM 9. Certificate of Interest

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

DataTern, Inc. v. Blazent, Inc., et al.

No. 13-1251, 13-1252

CERTIFICATE OF INTEREST

Counsel for the (petitioner) (appellant) (respondent) (appellee) (amicus) (name of party) Appellant, DataTern, Inc. certifies the following (use "None" if applicable; use extra sheets if necessary):

1. The full name of every party or amicus represented by me is:

DataTern, Inc.

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by me is:

DataTern, Inc.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party or amicus curiae represented by me are:

Amphion Innovations, PLC

4. ☐ The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this court are:

McCarter & English, LLP, Lee Carl Bromberg, Erik Belt, William A. Zucker

3/29/13
Date

Signature of counsel

Erik Paul Belt
Printed name of counsel

Please Note: All questions must be answered

cc: Counsel of Record

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

STATEMENT OF RELATED CASES

This appeal is related to Federal Circuit Appeal Nos. 13-1184 and 13-1185, both of which involve the same appellant and same patent as this appeal. Specifically, in *Microsoft, Inc. v. DataTern, Inc.* (No. 13-1184), DataTern, Inc., appeals a judgment of non-infringement based on erroneous claim construction of U.S. Pat. No. 6,101,502 ("the '502 patent").[1] Likewise, in *SAP AG et al. v. DataTern*, *Inc*. (No. 13-1185), DataTern also appeals a judgment of non-infringement of the '502 patent.

The present appeal involves two cases from the District of Massachusetts, *DataTern, Inc*. *v. Microstrategy*, *Inc*., Case No. 1:11-cv-12220-FDS (Appeal No. 13-1252), and *DataTern, Inc. v. Blazent*, *Inc*., Case No. 1:11-cv-11970-FDS (Appeal No. 13-1251), which were consolidated in the district court and for this appeal.

[1] In addition to the '502 patent, the *Microsoft* appeal also involves a second DataTern patent, U.S. Pat. No. 5,937,402 ("the '402 patent"). The '402 patent is not at issue in this appeal.

JURISDICTIONAL STATEMENT

The two underlying district court cases, which were consolidated, involve infringement of a United States patent and, accordingly, the district court had subject matter jurisdiction under 28 U.S.C. § 1338(a).

This appeal follows final judgments of non-infringement in the two underlying district court cases entered on February 7, 2013. A1, A5. Plaintiff-Appellant DataTern, Inc., timely filed its notices of appeal on March 5, 2013. A226, A1134. This Court has appellate jurisdiction under 28 U.S.C. § 1295(a)(1).

THE ISSUE PRESENTED FOR REVIEW

Whether the judgment of non-infringement should be reversed because it is based on a construction of the claim term "to create" that limits this ordinary English verb to a specific technical method of creation, even though neither the claim wording nor the specification requires or supports such a limitation.

STATEMENT OF THE CASE

I. THE PATENT-IN-SUIT

This case concerns the infringement of DataTern's U.S. Pat. No. 6,101,502 ("the '502 patent"). The '502 patent is directed to methods for enabling software applications written in object-oriented code to access data stored in relational databases. Briefly, in the late 1990s, when the claimed methods were invented, the

software industry was experiencing a problem known as the "object-relational mismatch." A1071 at ¶ 8. That is, software written in the most popular form of programming (so-called "object oriented" programming) could not easily access data stored in the most popular form of computer database (the so-called "relational database") because the organizational structures of each differed. The '502 patent solved this problem by introducing, in essence, the use of an intermediary between the application software and database software. By using this invention, the database could be changed, and data could be accessed, without requiring the application to be rewritten, thus increasing efficiency. A1074 at ¶ 16.

In particular, the '502 patent introduced a data services layer that includes a "runtime engine," which uses go-betweens known as "interface objects" to access data from the relational database. A243 at 1:64–66; *see also* A1074 at ¶ 17. The patent summarizes the invention as follows:

> In accordance with the present invention, a mapping between an object model and a relational database and a runtime engine are employed to facilitate access to a relational database . . . The database schema, object model, and mapping are employed to provide interface objects that are utilized by an object oriented software application to access the relational database . . . The interface object and runtime engine perform read and write operations on the database.

A243 at 1:53–66.

The happy result of this innovation is that "neither programmers nor software applications need have knowledge of the database structure, the database

programming interface, database security, or the database transaction model in order to obtain access to the relational database." *Id*. at 1:66–2:3.

In keeping with the teaching of the '502 patent, Claim 1 (with the disputed term bolded) provides as follows:

> 1. A method for interfacing an object oriented software application with a relational database, comprising the steps of:
>
> selecting an object model;
>
> generating a map of at least some relationships between schema in the database and the selected object model;
>
> employing the map **to create at least one interface object** associated with an object corresponding to a class associated with the object oriented software application; and
>
> utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

A246 at 7:53–8:3.

II. THE UNDERLYING LITIGATIONS

In December 2011, DataTern sued MicroStrategy, Inc., in the United States District Court for the District of Massachusetts, Boston Division, for infringement of the '502 patent. A229. In two sets of infringement contentions filed in April and July 2012, DataTern documented how MicroStrategy infringes various claims of the '502 patent by making, using, selling, and/or offering to sell Microstrategy's Business Intelligence software platform and related or derivative products and services. *See* A294 and A1087.

At roughly the same time, in November and December 2011, DataTern also sued eight other defendants in the District of Massachusetts (Carl Warren and Company, Inc.; Lancet Software Development, Inc.; Airlines Reporting Corp.; Informatica Corp.; Magic Software Enterprises Ltd. and Magic Software Enterprises, Inc.; Teradata Corp.; Epicor Software Corp.; and Premier, Inc.). These other eight defendants are customers of MicroStrategy and use or resell MicroStrategy's accused Business Intelligence software platform. As seen in DataTern's infringement contentions, these other defendants (dubbed the "Customer Defendants" by the district court) use, make, sell, or offer to sell products and services that are built on or incorporate MicroStrategy's Business Intelligence platform or key components of it. *See* A328, A372, A408, A440, A475, A517, 1549, A612, A674, A736, A788, A837, A888, A946, and A998.

Shortly after these cases were filed, the district court consolidated the suit against MicroStrategy with the suits against the Customer Defendants, designating MicroStrategy as the lead defendant under case number 11-cv-12220-FDS. A261. Because the Customer Defendants agreed to be bound by any judgment against MicroStrategy, the district court stayed discovery and other deadlines involving the Customer Defendants in favor of the lead case against MicroStrategy. A261; A1082. Thus, if the accused MicroStrategy software platform infringes, the Customer Defendants will also be liable.

In November 2011, DataTern also filed suit in the District of Massachusetts alleging that Defendant-Appellee Blazent, Inc., infringes the '502 patent (No. 11-cv-11970-FDS). A49. Blazent is also a MicroStrategy customer and uses, or bases its accused products and services on, Microstrategy's accused software platform. A69. In January 2013, the district court consolidated the *Microstrategy* case with the earlier-filed *Blazent* case. A47.[2] As with the Customer Defendants, Blazent will be liable for infringement if Microstrategy is found to infringe.

While the Boston cases were proceeding, DataTern was defending itself in two earlier-filed declaratory judgment actions in the Southern District of New York. These two cases also involved the '502 patent as well as one other DataTern patent not at issue here, U.S. Pat. No. 5,937,402 ("the '402 patent"). Specifically, on April 7, 2011, Microsoft, Inc., sued DataTern for a declaratory judgment of non-infringement or invalidity of the '502 and '402 patents. Less than two weeks later, SAP AG and SAP America, Inc. (collectively, "SAP") filed a similar declaratory judgment action against DataTern, also in the Southern District of New York and also involving the '502 and '402 patents. Both the *Microsoft* and *SAP* cases (collectively, the "New York cases") are the subject of pending appeals to this Court. *See* Appeal Nos. 13-1184 and 13-1185.

[2] For purposes of this brief, the consolidated *Blazent* and *Microstrategy* cases will be referred to as the "Boston cases."

In July 2012, the New York court held a joint *Markman* hearing in the New York cases. In August 2012, the New York court issued a claim construction order interpreting various disputed terms of the '502 and '402 patents. *See* A101. Recognizing that, under the New York court's construction, DataTern could not prove infringement, DataTern offered to concede infringement as the most efficient means of proceeding. Based on DataTern's concessions, the district court granted judgment of non-infringement, A134, and then entered final judgment. A142. The appeals of the New York cases soon followed.

The Boston court indicated that, in the interest of judicial efficiency, it would follow the New York court's claim construction as to the '502 patent. A6. Relevant to this appeal, and as discussed more extensively in the next section, the New York court construed the claim term "to create at least one interface object" to mean, in relevant part, "to generate code for at least one class and instantiate an object from that class." A132.

Based on this portion of the New York court's construction, DataTern conceded infringement in the Boston cases. A73. Specifically, DataTern conceded that it could not show that, in order to create an interface object on each occasion, the MicroStrategy Business Intelligence platform generates code for a class and instantiates an object from that class, as required under the New York court's claim construction. A77. Applying the New York court's construction,

and in light of DataTern's concession, the Boston court granted judgment of non-infringement on February 7, 2013. That judgment is based solely on DataTern's one concession regarding "to create at least one interface object." A1; A5. Microstrategy had moved for summary judgment of non-infringement based on other grounds, but the Boston court rejected those other grounds as moot. A3; A7. DataTern noticed this appeal on March 5, 2013. A226; A1134.

III. THE CLAIM CONSTRUCTION DISPUTE

The only issue in this case is the construction of "to create at least one interface object," as that phrase appears in Claim 1 of the '502 patent. The New York court construed the phrase to mean "to generate code for at least one class and instantiate an object from that class, where the object is not part of or generated by the object-oriented application and is used to access the database." A132 (emphasis added). Only the underlined part of the district court's construction is relevant to this appeal. As will be argued below, the applied construction of "to create at least one interface object" is wrong because it improperly reads a limitation from a figure in the specification into the claim and, in any event, is neither consistent with nor supported by the intrinsic evidence.

The table below compares the New York court's construction of "to create at least one interface object" with DataTern's proposed construction:

DISPUTED CLAIM TERM "employing a map **to create at least one interface object** associated with an object corresponding to a class associated with the object oriented software application;	
The New York Court's Construction Applied by the Boston Court	**DataTern's Proposed Construction**
"To generate code for at least one class and instantiate an object from that class, where the object is not part of or generated by the object-oriented application and is used to access the database"	The phrase has its ordinary meaning to one of ordinary skill having read the '502 patent and prosecution history. No further construction is necessary. To the extent that further definition is necessary, however, than the following terms have the following meanings: "To create" means "to make." An "Interface Object" means an "object by which the object oriented application accesses the relational database via a runtime engine."

A116-117; A132.

In the New York cases, DataTern argued that "to create" has its ordinary meaning (*i.e.*, synonymous with "to make") and thus no further construction was needed. DataTern further argued that the wording of Claim 1 required only that a map be employed in the creation step ("employing a map to create at least one interface object . . .") and that, with this exception, nothing else in the claims or

specification of the '502 patent requires that "to create at least one interface object" be practiced through any particular method of creation.

Neither the court nor the parties cited any evidence that "to create" is a technical or esoteric word requiring clarification for a lay jury. Neither the court nor the parties cited any intrinsic evidence suggesting that "to create" requires any further construction. Rather, the New York court felt compelled to interpret, and ultimately narrow, the term simply because "[t]he parties do not agree on the method of the creation." A117.

In construing the term, the New York court relied on Figure 1 of the '502 patent, which illustrates certain alternative embodiments of the claimed invention. *See* A118. Based on Figure 1 and related attorney argument, the New York court found that this term requires code generation:

> Figure 1 therefore requires code generation to be a necessary part of the creation of interface objects. While Figure 1 is only one embodiment of the invention, defendant's own expert conceded that he is not aware of any embodiments of the '502 Patent that do not require code generation.

Id.

As argued below, nothing in the '502 patent limits the claims to the method required by the New York court. Further, the New York court apparently misunderstood or ignored the expert testimony, which did not concede that the claimed creation step is limited to code generation.

Finally, the New York court, apparently, ignored the heavy presumption that claim terms have their ordinary meaning unless clearly and expressly disclaimed in the specification or prosecution history. The New York court failed to point to any such disclaimer, and there is none. The New York court also failed to apply other well-established canons of claim construction, such as, for example, the doctrine of claim differentiation. While Claim 1 recites merely "to create at least one interface object," Claim 10 recites "a code generator that employs said map to create at least one interface object." Claim 1 says nothing about a "code generator" or "generating" code. The New York court nonetheless rejected the claim differentiation argument solely due to "other language differences between Claims 1 and 10 that further differentiate them." A118–19. As argued below, that reasoning is also erroneous.

As seen above, the Boston court applied the New York court's construction and entered judgment of non-infringement against DataTern. If the New York court had construed "to create" to have its ordinary meaning of "to make," as DataTern proposed, then DataTern could have proved infringement of the '502 patent by the accused Microstrategy Business Intelligence platform and related or derivative products and services. *See* A1087.

STATEMENT OF FACTS

I. BACKGROUND OF THE '502 PATENT

The '502 patent is entitled "Object Model Mapping and Runtime Engine for Employing Relational Database with Object Oriented Software." As its title suggests, the '502 patent is directed to methods and systems for enabling object-oriented software applications to access data stored in relational databases. Before the '502 patent, the inability of object-oriented applications to interact efficiently with relational databases had been a vexing problem in the software industry. In effect, the '502 patent taught ways of facilitating those interactions between object-oriented applications and relational databases.

Employees of Ontos, Inc., invented the claimed subject matter of the '502 patent. A1067 at ¶ 3. The '502 patent was the product of substantial investments and development efforts by highly skilled software specialists and engineers. *Id*. at ¶ 4. In 2001, Ontos assigned the '502 patent to FireStar, Inc., its sister company. FireStar's business was the development of software products, and FireStar used the inventions of the '502 patent in connection with a software product known as "ObjectSpark." *Id*. at ¶ 5. Both Ontos and FireStar were backed by Amphion Innovations plc, which funds new and innovative companies in, for example, the information technology and medical devices fields. A1066 at ¶ 2.

FireStar, however, could not pursue both the development of software products and the enforcement of its patent rights against increasing third-party infringements. A1067 at ¶¶ 6–7. Accordingly, in 2007, FireStar assigned the '502 patent to DataTern, a subsidiary of Amphion. A1066 at ¶ 1 and A1067 at ¶ 4. DataTern assumed responsibility for the licensing and enforcement of the '502 patent, freeing FireStar to pursue innovation and sales of its software products. Licensee fees for use of the '502 patent paid to DataTern become part of the funds that are used to support FireStar and Amphion's other innovative, venture-backed companies. A1067 at ¶ 4.

II. THE INNOVATION OF THE '502 PATENT

At the time of the invention, object-oriented programming had replaced procedural programming as the predominant method for writing enterprise software and other complex software code. A1071 at ¶ 9. Examples of object-oriented programming languages contemporaneous with the '502 patent include Java, C++, and SmallTalk. *Id*. Programs written in an object-oriented programming language are structured such that in-memory instances of objects interact with one another to perform required tasks. *Id.* An object-oriented program may use methods on these object instances without knowledge of the internal structure or inner workings of the object instances. *Id.* Thus, object-

oriented programming techniques increased the efficiency of writing and maintaining computer programs. A1073 ¶ 13.

The most popular form of a database is known as a relational database. Examples of relational databases contemporaneous with the '502 patent include IBM DB2, Oracle Database, and Microsoft SQL Server. A1072 at ¶ 11. A relational database stores data in tables known to those of ordinary skill in the art as "physical tables." *Id.* Physical tables present rows of data collected in one or more columns and can be visualized as presenting data in much the same way as data is presented in a spreadsheet. *Id.* The organizational structure of a relational database is often referred to as the "schema" of the database. *Id*. at ¶ 12.

Due in part to the different goals of object-oriented programming and relational database design, the structure of objects for an object-oriented program is often different from the table structure of the relational database schema. A1073 at ¶ 13. This problem became known as the "object-relational mismatch." *Id.* Due to this mismatch, enabling object-oriented applications to access data from relational databases was a potentially time-consuming and labor-intensive process fraught with human error. *Id*. at ¶ 14.

The inventors of the '502 patent recognized object-relational mismatch as a problem in the software industry. As the '502 patent specification explains, the problem resulted in inefficiencies and errors:

> The need for interfacing object oriented software applications with relational databases is well known. One method of interfacing an object oriented application with a relational database is to adapt the requests made by the application to the relational database. More particularly, object operations are translated into relational database queries. However, this technique is processor intensive and sacrifices some of the advantages associated with the object oriented model. As a result, the object-oriented software application is unable to function efficiently.
>
> Another method of interfacing an object oriented application with a relational database is to translate database information into a format which is compatible with the object oriented application. Relational databases typically separate data into a plurality of tables through a process known as "normalization" to minimize duplication. A normalized relational database includes a plurality of tables, wherein each table includes at least one field and one key, and at least one field in each table is uniquely dependent upon the key that is associated with the table. These tables can be translated into objects. However, the objects can become inaccurate when changes are made to the relational database. It is known to adapt to changes in the database by performing further translations, but this process requires substantial effort.

A243 at 1:20-50 (emphasis added).

The '502 patent solves this object-relational mismatch and its symptoms by the use of certain mechanisms, including a runtime engine and at least one interface object, to allow the object oriented application to communicate with the relational database without the need for customizing one for the other:

> The present invention provides transparent access to the relational database. The interface objects and runtime engine perform read and write operations on the database, including generation of SQL code. Consequently, neither programmers nor software applications need have knowledge of the database structure, the database programming interface, database security, or the database transaction model in order

> to obtain access to the relational database. Further, changes to the relational database do not always necessitate additional mapping.

Id. at 1:63–2:5; *see also* A1074 at ¶ 15.

This invention further allows software developers to independently optimize the object oriented program and the relational database for their respective purposes without having to take into consideration how the changes to one will impact the other. A1074 at ¶ 17. By providing an intermediary between the object oriented application and the database, the database may be changed without requiring the application to be recoded, thus increasing computing efficiency. *Id*. at ¶¶ 16–17. Thus, the '502 patent claims inventions that advance computer science and database technology.

The invention of the '502 patent provides an elegant solution to the object-relational mismatch problem. Claim 1 of the '502 patent, for example, provides (with the disputed claim phrase bolded):

> 1. A method for interfacing an object oriented software application with a relational database, comprising the steps of:
>
> selecting an object model;
>
> generating a map of at least some relationships between schema in the database and the selected object model;
>
> employing the map **to create at least one interface object** associated with an object corresponding to a class associated with the object oriented software application; and
>
> utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

A246 at 7:53–8:3 (emphasis added). Using the method of Claim 1, a software programmer can readily provide an interface between an object oriented software application and a relational database.

The method of Claim 1 begins with the selection of an object model. The '502 patent explains that an object model is "a template that has a predetermined standardized structure." A243 at 2:40–41. Although the object model is not the same as the object-oriented application, the object model may be a template of business data useful to the application. *See*, *e.g.*, A237 (Figure 1 separately illustrating "object model 14" and "object oriented application 22") and A246 at 7:53–8:3 (Claim 1 separately reciting "an object oriented software application" and "an object model"). The exemplary object model 14 illustrated in Figure 3 "includes attributes and inheritances relationships that are mapped to relational database features . . . ." A243 at 2:41–44. Accordingly, the second step of Claim 1 is the generation of a map of at least some relationships between schema in the database and the selected object model.

Figure 1 of the '502 patent is a block diagram that illustrates the use of the map to create interface objects that are employed by a runtime engine and an object oriented application to access a relational database. *See id*. at 2:13–16. As illustrated below, Figure 1 shows alternative means for creating interface objects:

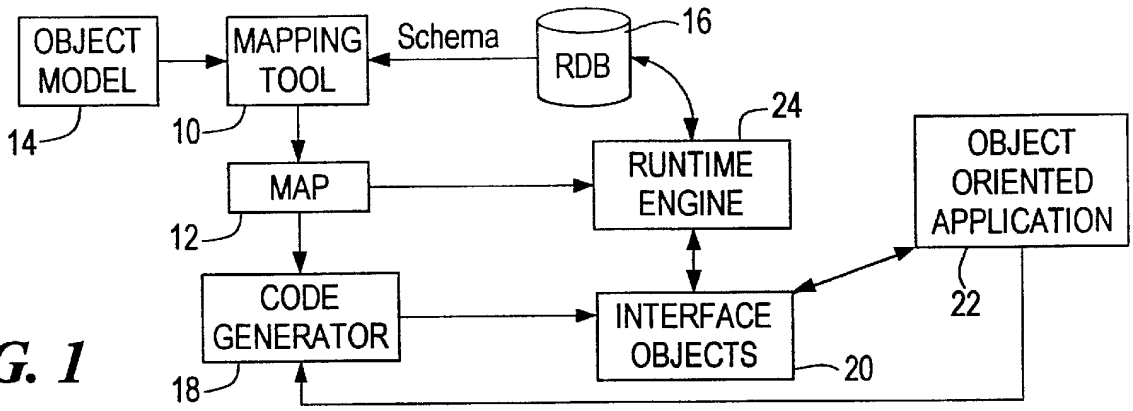


FIG. 1

Figure 1 illustrates a map 12, a code generator 18, and interface objects 20 with one arrow leading from the map 12 to the code generator 18 and another arrow leading from the code generator 18 to interface objects 20. Figure 1, however, also illustrates a map 12, a runtime engine 24, and interface objects 20 with one arrow leading from the map 12 to the runtime engine 24 and another arrow leading from the runtime engine 24 to interface objects 20. Thus, Figure 1 indicates that either the runtime engine 24 or the code generator 18 could use the map 12 to create interface objects 20.

Other figures show that the creation process is not constrained to generating code for a class. Figure 7, for example, is a sequence diagram that illustrates operation of the runtime engine. *See* A243 at 2:23–24. Figure 7 illustrates an exemplary creation of a data services layer object ("DslObject ") by a "business object." Specifically, Figure 7 "illustrates the sequence of actions that take place when a business object creates a Dsl object in step 61, accesses the name property

in step 65[,] and saves the object in step 69." A245 at 6:31–44 (emphasis added). Neither Figure 7 nor the related description indicate that the business object is a code generator or that the process involves the generation of code for at least one class, as the New York court required in its claim construction ruling.

The '502 patent discloses embodiments of the invention in which other constructs besides interface objects are created without using the specific technical method dictated by the New York court's claim construction. For example, the patent teaches the optional creation of an object model and of database schema. *See* A243 at 1:56–58; *see also* Abstract (referencing the creation of an object model and the creation of database schema). Further, Claim 41, which depends from Claim 1, requires the step of "creating schema in the [relational] database" in addition to the step of "employing the map to create at least one interface object." A252. Similarly, Claim 39 depends from Claim 1 and requires the step of "creating the object model" in addition to the step of "employing the map to create at least one interface object." *Id*. The '502 patent uses the verb "to create" with respect to a variety of constructs, thus evidencing a broad, generic meaning of "to create."

III. THE ACCUSED PRODUCTS AND SERVICES

As prefaced above, DataTern accuses MicroStrategy's Business Intelligence platform, as well as related products and services, of infringing various claims of

the '502 patent.[3] As summarized below, DataTern's infringement contentions describe how MicroStrategy's Business Intelligence platform enables an object oriented software application to interact with, and access data from, a relational database in accordance with the limitations of the asserted claims.

The accused MicroStrategy Business Intelligence platform implements a method for interfacing an object oriented software application with a relational database. A1091. As part of this method, the MicroStrategy Business Intelligence platform selects an object model. A1094. A map is generated of at least some relationships between schema in the database and the selected object model. A1096. The MicroStrategy Business Intelligence platform employs the map to create at least one interface object that is associated with an object corresponding to a class associated with the object oriented software application (*e.g.*, the reporter module of the MicroStrategy Business Intelligence or with a third party application). A1097. Finally the runtime engine of the MicroStrategy Business Intelligence platform is utilized in invoking the interface object with the object oriented application to access data from the relational database. A1100

[3] For example, DataTern accuses products and services used, made, sold, or offered by Blazent and the Customer Defendants that are based on MicroStrategy's Bustiness Intelligence platform. Because these other defendants have agreed to be bound by any judgment against MicroStrategy, however, the Court need only consider MicroStrategy's accused software platform.

Although the Business Intelligence platform certainly creates and uses interface objects, A1097, DataTern conceded that the accused products do not create the interface objects according to the specific technical method dictated by the New York court's claim construction order. That is, as DataTern conceded, the MicroStrategy Business Intelligence platform does not "generate code for at least one class" from which the interface object is instantiated each time an interface object is created. A2 at ¶ 2.

SUMMARY OF ARGUMENT

This appeal concerns the meaning of the claim term "to create." This common English verb is readily understandable to a lay jury and requires no further construction. Nonetheless, at the urging of DataTern's opponents in the related New York cases, the New York court limited this ordinary English word by giving it additional meaning not required by the patent. The Boston court applied the New York Court's construction. Based on this construction, DataTern conceded non-infringement in the New York cases and, likewise, in the Boston cases, which are the subject of this appeal. But for the New York court's construction, DataTern could have proved infringement against MicroStrategy and the other Boston defendants.

The construction at issue was reached only by disregarding well-established canons of claim construction. First, the district court failed to apply the heavy

presumption that the words of the claim carry their ordinary meaning. This presumption holds special force when the word is an ordinary English word like "create." The word is neither a term of art, a technical word, nor some other esoteric word that would require clarification for a lay jury. Moreover, nothing in the claim wording or other intrinsic evidence even suggests that the word is unclear or has some special meaning and that would require interpretation. Unless a patent points to some intrinsic reason for construing an ordinary word like "create," it should be left alone.

Instead of presuming that the term has its ordinary meaning and thus requires no further construction, the district court appears to have presumed the opposite—*i.e.*, that the term required construction and was necessarily limited to a particular technical method disclosed in one of the figures of the '502 patent. At the same time, the district court improperly ignored the '502 patent's use of the verb "create" with respect to other constructs that could not be created by the same method as an interface object. Particularly because the '502 patent uses the verb "create" in the sense of its widely accepted meaning to indicate the use of any of a broad range of possible methods--and not to indicate the application of a special technical method—the district court's unduly narrow construction must be rejected for failure to follow the proper claim construction paradigm. In short, both the claim wording and specification support the ordinary English meaning of "to

create. Nothing in the claims nor the specification supports the additional limitations imposed by the district court.

ARGUMENT

The judgment of non-infringement rests on faulty claim construction. Claim construction is an issue of law that this Court reviews *de novo*. Likewise, this Court reviews the grant of summary judgment without deference. *Arlington Indus., Inc. v. Bridgeport Fittings, Inc*., 632 F.3d 1246, 1252–53 (Fed. Cir. 2011).

The only issue for this Court's *de novo* review is the construction of "to create at least one interface object," which the district court interpreted to mean "to generate code for at least one class and instantiate an object from that class." A132. But for this construction, DataTern could have proved that the accused products and services infringe one or more claims of the '502 patent.

I. THE CLAIM WORDING SHOWS THAT "TO CREATE" IS A BROAD TERM NOT LIMITED TO ANY ONE TECHNICAL METHOD

Claims alone define patented inventions. Thus, claim construction begins and ends with the words of the claims. *Phillips v. AWH Corp*., 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc); *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996) ("First, we look to the words of the claims themselves, both asserted and nonasserted, to define the scope of the patented invention").

Courts must indulge a "heavy presumption" that the words of the claim carry their ordinary meaning. *Teleflex, Inc., v. Ficosa North America Corp*., 299 F.3d 1313, 1325 (Fed. Cir. 2002) (citation omitted).

After forming an understanding of the words of the claims, a court may consult the specification and prosecution history to confirm its understanding and to see whether the inventor has given special meaning to the terms. *Vitronics*, 90 F.3d at 1582. Although the patent specification and prosecution history are consulted, "[t]he written description part of the specification itself does not delimit the right to exclude. That is the function and purpose of claims." *Markman v. Westview Instruments, Inc*., 52 F.3d 967, 980 (Fed. Cir. 1995), *aff'd*, 517 U.S. 370 (1996); *see also Phillips,* 415 F.3d at 1314 ("the claims themselves provide substantial guidance as to the meaning of particular claim terms"). Thus, a disputed term like "to create" cannot be restricted to a particular embodiment in the specification unless the inventor has unambiguously defined the term or has clearly disclaimed a broader scope. *Teleflex Inc.,* 299 F.3d. at 1328; *see also Omega Eng'g, Inc. v. Raytech Corp.*, 334 F.3d 1314, 1329 (Fed. Cir. 2003) ("It is axiomatic that, unless expressly compelled by the intrinsic evidence, courts must avoid the addition of a novel limitation"); *Dayco Prods., Inc. v. Total Containment, Inc.*, 258 F.3d 1317, 1327 (Fed. Cir. 2001) ("adding limitations to claims not

required by the claim terms themselves, or unambiguously required by the specification or prosecution history, is impermissible").

A. The Plain English Verb "To Create" Is Broad

Claim 1 requires the step of "employing the map to create at least one interface object." The wording of Claim 1, as well as of other claims, shows that "to create" must be interpreted broadly.

First, "to create" is an ordinary English verb and thus requires no further construction. Certain terms may be ordinary English, non-technical words and thus readily understandable to anybody. Such words typically require no further construction. *See In re Wright*, 866 F.2d 422, 425 (Fed. Cir. 1989) ("These are common, garden variety words known to every English-speaking person"); *see also, e.g., ActiveVideo Networks, Inc. v. Verizon Comm'ns, Inc.*, 694 F.3d 1312, 1325-26 (Fed. Cir. 2012) (district court correctly declined to construe ordinary English word, such as "superimposing" and "including," which had their plain and ordinary meanings); *Desper Prods., Inc. v. QSound Labs, Inc.*, 157 F.3d 1325, 1336 (Fed. Cir. 1998) ("Common words . . . should be interpreted according to their ordinary meaning"); *CollegeNET, Inc. v. Xap Corp.*, No. CV-03-1229, 2004 U.S. Dist. LEXIS 22370, at *42 (D. Or. Oct. 29, 2004) (declining to construe "creating," which has its ordinary English meaning).

To the extent any further construction is necessary, "to create" is synonymous with the verb "to make," although a jury would readily understand that meaning in any event. Indeed, requiring a court to construe plain, ordinary English terms wastes judicial resources and, further, serves only to confuse a jury by adding more words and layers that need to be deciphered. *See U.S. Surgical Corp. v. Ethicon, Inc.*, 103 F.3d 1554, 1568 (Fed. Cir. 1997) (claim construction is required only "when the meaning or scope of technical terms and words of art is unclear and in dispute and requires resolution to determine" the issue at hand).

Second, Claim 1 does not modify "to create" or otherwise specify how the interface object is created (except, of course, that a map is employed in the process). The district court did not point to a single word in Claim 1 that invites further construction of "to create" or suggesting that further construction is needed. And there is none. Thus, Claim 1 leaves no opening for reading in further limitations, whether from embodiments illustrated in the specification or from extrinsic evidence. *See Johnson Worldwide Assocs., Inc. v. Zebco Corp*., 175 F.3d 985, 989–90 (Fed. Cir. 1999) ("claim terms cannot be narrowed by reference to the written description or prosecution history unless the language of the claim invites reference to those sources").

Third, the use of "create" in other claims also shows how broadly it must be understood. As discussed below, the verb "create" appears throughout the claims

in connection with a variety of creation steps. This creation step should carry the same meaning in each instance. *See Paragon Solutions, LLC v. Timex Corp*., 566 F.3d 1075, 1087 (Fed. Cir. 2009) ("the same terms appearing in different portions of the claims should be given the same meaning unless it is clear from the specification and prosecution history that the terms have different meanings at different portions of the claims"). But applying the district court's construction of "to create" to every other occurrence of the term in the claims results in absurdities, thus showing that the district court's interpretation is incorrect.

For example, Claim 41 incorporates the step from Claim 1 of "employing the map to create at least one interface object" and adds the step of "creating schema in the database." A252 (emphasis added). Thus, Claim 41 requires the creation of both an interface object and database schema. The schema in the database is its organizational structure. *See* A1072 at ¶¶ 11, 12 ("The organizational structure within a relational database is referred to as the schema of the database"). The organizational structure of a database, however, cannot be "created" via the method dictated by the district court's construction. In other words, under the district court's construction--and given the presumption that the same term appearing in different portions of the claims should be given the same meaning–code generation and instantiation of an object would be imported into the meaning of "creating" recited in Claim 41. But neither the generation of code for a

class nor the instantiation of an object of a class is relevant to the creation of the organizational structure of a database. Indeed, the district court's construction of "create" can result only in oriented-oriented programming constructs limited by a particular computer programming language convention. (And the '502 patent is not limited to any one computer programming language.) It is this very difference between database schema and object-oriented program constructs that gives rise to the "object-relational mismatch" problem solved by the '502 patent. *See* A1073–74 at ¶¶ 13–15 (stating the '502 patent provides a solution to the problem of automating or simplifying the interaction between object-oriented programs and relational databases). In short, because Claim 41 uses "create" with respect to two very different constructs (interface objects and database schema), and given the presumption that "create" should carry a consistent meaning throughout, Claim 41 shows that "create" is a broad term not limited to the particular method of creation required by the district court's construction.

Like Claim 41, Claim 39 depends from Claim 1 and adds an additional creation step: namely, "creating the object model." A252. Thus, Claim 39 requires the creation of both an interface object and an object model. But under the district court's construction of "object model," an object model could not be "created." Indeed, an object model includes no object instantiated from a class, as required by the district court's construction. *See, e.g.*, A238 at Fig. 3 (exemplary

object model 14 including no "object") and A132 (under district court's construction, object not included in the definition of "object model"). Like Claim 41, Claim 39 shows that the '502 patent uses "create" as a broad term that is not limited to the particular method of creation required by the New York court's construction. *See*, *e.g.*, *Cordis Corp. v. Medtronic AVE, Inc.*, 339 F.3d 1352, 1356–57 (Fed. Cir. 2003) ("Nothing about the phrase 'slots formed therein' suggests that the slots must be formed by a particular process; the phrase certainly does not indicate that the wall must be formed first and the slots formed later").

B. The Doctrine of Claim Differentiation Further Supports DataTern's Reading

In contrast to Claim 1, Claim 10 specifically recites the limitation of a "code generator." Claim 10 recites, in relevant part, "a code generator that employs said map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application." A247. Under the doctrine of claim differentiation, the limitation of a "code generator" from Claim 10 should not be imported into Claim 1. *See Clearstream Wastewater Sys., Inc.v. Hydro-Action, Inc*., 206 F.3d 1440,1446–47 (Fed. Cir. 2000); *Tandon Corp. v. U.S. Int'l Trade Com'n* 831 F.2d 1017, 1023 (Fed. Cir. 1987) ("There is presumed to be a difference in meaning and scope when different words or phrases are used in separate claims").

The district court erroneously rejected this claim differentiation argument on the mistaken belief, apparently, that claim differentiation does not apply when there are also other "language differences between Claims 1 and 10." A118–19. But this Court has held that application of the claim differentiation doctrine is not limited to comparisons between an independent and its dependent claim. Rather, the comparison can also be made between two independent claims. *Hologic, Inc. v. SenoRx, Inc*., 639 F.3d 1329, 1336 (Fed. Cir. 2011). Although there may be other differences, a comparison between Claim 1, which does not specify a structure or specific technical method for creating interface objects, and Claim 10, which does, shows that code generation cannot be read into Claim 1 to limit "to create at least one interface object."

II. THE SPECIFICATION DOES NOT LIMIT "TO CREATE"

The heavy presumption that a claim term carries its plain and ordinary meaning cannot be overcome simply by pointing to the preferred embodiment or other structures or steps disclosed in the specification or prosecution history. *See CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002). Even if the specification discloses a single embodiment, the claim cannot be confined to that embodiment unless the inventor has clearly disclaimed other embodiments. *See Arlington Indus., Inc. v. Bridgeport Fittings, Inc*., 632 F.3d 1246, 1254 (Fed. Cir. 2011); *Liebel-Flarscheim Co. v. Medrad, Inc*., 358 F.3d 898, 906 (Fed. Cir.

2004); *SRI Int'l v. Matsushita Elec. Corp. of America*, 775 F.2d 1107, 1121 n.14 (Fed. Cir. 1985) ("That a specification describes only one embodiment does not require that each claim be limited to that one embodiment").

The New York court inferred from Figure 1 of the '502 patent that Claim 1 must somehow require code generation. A118. It is true that Figure 1 illustrates one possible embodiment in which code generation is used to create the interface object. For instance, as seen below, Figure 1 illustrates a map 12, a code generator 18, and interface objects 20 with arrows leading from the map 12 to the code generator 18 and then to interface objects 20. The district court relied on these arrows to limit "to create" to the use of code generation. *Id*.

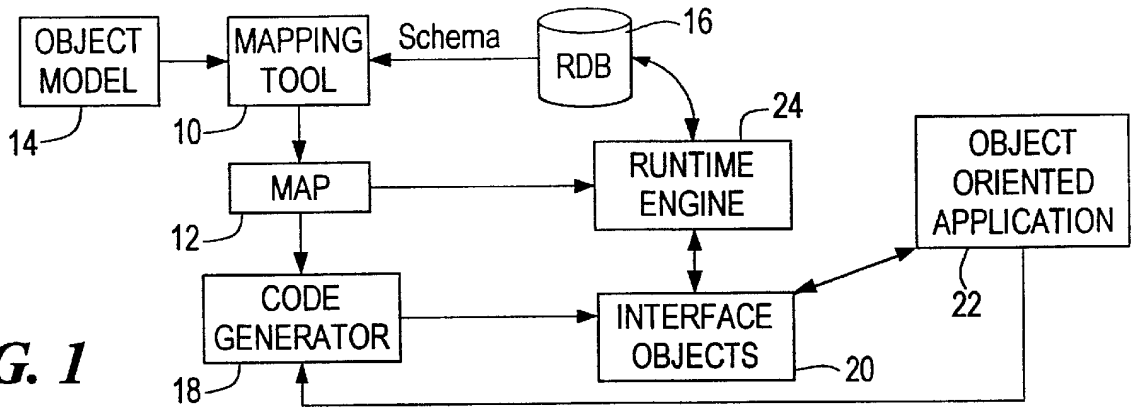


FIG. 1

Figure 1, however, also illustrates an embodiment that does not rely on code generation. Figure 1 illustrates a map 12, a runtime engine 24, and interface objects 20 with arrows leading from the map 12 to the runtime engine 24 and then to interface objects 20. Thus, Figure 1 equally shows that the runtime engine could

also employ the map to generate interface objects. But even if Figure 1 did not show this alternative embodiment, and even if Figure 1 could be read to illustrate a single creation step that relies solely on code generation, then it still would have been error for the district court to confine the interface object creation step to that one particular illustrated method. *See Arlington Indus*., 632 F.3d at 1254 ("While the drawings of the adaptor consistently depict an incomplete circle, drawings in a patent need not illustrate the full scope of the invention").

The New York court concluded that Figure 1 "requires code generation to be a necessary part of the creation of interface objects," even though this conclusion has no basis in the claim wording and is contrary to what the court acknowledged is actually illustrated in Figure 1. *See* A118 ("Figure 1 does allow for the map to bypass the code generator and go directly into the runtime engine"). Moreover, the court based its understanding of Figure 1 on argument by Microsoft and SAP's counsel. *Id*. (citing transcript of attorney argument at *Markman* hearing).

In an attempt to bolster its improper reliance on attorney argument over intrinsic evidence, the New York district court reasoned that "[DataTern's] own expert conceded that he is not aware of any embodiments of the '502 patent that do not require code generation." *Id*. This reliance on the expert's testimony is flawed for two reasons.

First, the district court apparently misunderstood or ignored the expert testimony. The cited expert explained that "[i]nterface objects may be instantiated from pre-existing classes, or as in the preferred embodiment, from generated classes." A1075 at ¶ 20. In other words, it is not necessary to generate code for a class each time an object is created. Rather, an interface object may be created from a pre-existing class. In the New York cases, Microsoft and SAP's expert agreed that "[y]ou can think of a class as a template for a specific kind of object, or as a factory cranking out as many of its products as required." A111. Accordingly, neither expert testified that it is necessary to generate code for a class each time an object is created. Thus, it is hard to understand why the New York court concluded that the creation step in Claim 1 requires the generation of code for a new class.[4]

Second, as argued above, even if generating code for a class and instantiating an interface object from that class were the only embodiment shown, that would still not limit the claims absent some specific claim wording or clear and express disclaimer of scope in the specification. *See Rhine v. Casio, Inc.*, 183

[4] Moreover, DataTern's expert did not concede, as the New York court was led to believe, that code generation is required for creation. In the cited testimony, the expert was asked if he could identify any embodiments that use "anything but the code generator to generate interface objects?" A118 (emphasis added). The expert was not asked if he could identify any embodiments that use anything but a code generator to create interface objects. In other words, the expert was distinguishing between generation and creation.

F.3d 1342, 1346 (Fed. Cir. 1999) (reversing claim construction in which the district court confined the claim term at issue to the preferred embodiment, even though the claim did not recite the imported limitation and nothing in the written description required reference to that embodiment); *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d at 1366 (heavy presumption that a claim term carries its ordinary meaning cannot be overcome simply by pointing to the preferred embodiment or other structures or steps disclosed in the specification).

The New York district court also ignored the fact that the '502 patent uses the verb "to create" not just with respect to interface objects but also with respect to other structures. For example, the abstract of the '502 patent recites the creation of an object model and the creation of database schema. A236; *see also* A243 at 1:56–58. As explained above, if the district court's interpretation of "to create" were read into every other instance of "to create" (or "creating") in the patent, then neither an object model nor database schema could be created using the method required by the New York district court's construction. Accordingly, the use of the verb "to create" with respect to database schema and object model in the specification shows that the '502 patent as a whole uses the verb "to create" broadly, not to indicate a specific technical method of creation. As such, the

district court erred in restricting the verb to a specific technical method of creation.[5]

Figure 7 of the '502 patent also contradicts the district court's claim construction. According to the specification, Figure 7 "illustrates the sequence of actions that take place when a business object creates a Dsl object in step 61." A246 at 6:32-33 (emphasis added). Accordingly, in the foregoing sentence, the "business object" handles creation. Neither Figure 7 nor the related description indicates that the business object is a code generator or "generates code for at least one class," as required by the New York court's construction of "create." Thus, the specification again indicates that creation is not limited to the one particular method for creation of an interface object specified in the New York court's construction. Instead, the '502 patent uses the commonly understood verb "to create" as a broad term encompassing any method of creation consistent with its widely accepted meaning, "to make."

To recap, the New York court erred by not considering all of the intrinsic evidence related to the construction of "to create," by not applying the heavy presumption that the verb "to create" carried its plain meaning, and by electing to add limitations based solely on an exemplary embodiment. Thus, the New York district court's construction of "to create at least one interface object," which

[5] Indeed, giving the verb "to create" a hypertechnical construction, as the New York court did, is only going to invite confusion rather than clarity for a jury.

includes a very specific construction of the verb "to create," must be rejected as unduly narrow and inconsistent with the use of the verb "to create" in Claim 1 and in the '502 patent as a whole.

III. THE JUDGMENT OF NON-INFRINGEMENT IS BASED ON ERRONEOUS CLAIM CONSTRUCTION AND THUS MUST BE REVERSED

After the district court in the related New York cases entered judgment based on its claim construction order, DataTern conceded that the accused products in the Boston cases do not infringe the '502 patent if the New York district court's construction of the phrase "to create at least one interface object" is upheld. *See* A93 at ¶ 15; A94 at ¶ 18. DataTern conceded that the accused products—MicroStrategy's Business Intelligence platform and related or derivative products and services—do not "generate code for at least one class and instantiate an object from that class" in each instance, as required by the New York district court's construction. A94 at ¶ 19; A97 at ¶ 21. Applying the New York district court's construction and based on DataTern's concession alone, the Boston district court entered a judgment of non-infringement. *See* A6 at ¶¶ 2–7.

As explained above, the New York court's construction of "to create at least one interface object" must be rejected. DataTern has documented how the other limitations of the asserted claims of the '502 patent are met and, indeed, how the "to create at least one interface object" limitation is met under the correct

construction. *See generally* A290-1055 and 1077-1133. Because the judgment in the Boston cases is based solely on an erroneous construction, the cases should be remanded for further proceedings consistent with the proper construction.

CONCLUSION

For the reasons argued above, DataTern respectfully asks this Court to reverse the judgment of non-infringement and remand the case to the district court for further proceedings.

Respectfully submitted,

/s/ Erik P. Belt
Lee Carl Bromberg
Erik Paul Belt
William A. Zucker
Kia L. Freeman
Keith Toms
Gabriel Goldman
Amy J. Tindell
McCarter & English, LLP
265 Franklin Street
Boston, Massachusetts 02110
Telephone: (617) 449-6500

Attorneys for Defendant-Appellant

May 28, 2013

ADDENDUM

ADDENDUM

TABLE OF CONTENTS

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

DATATERN, INC., **Plaintiff,** **v.** **BLAZENT, INC.,** **Defendant.**	**Civil Action No. 11-11970-FDS**
DATATERN, INC., **Plaintiff,** **v.** **MICROSTRATEGY, INC., CARL WARREN AND COMPANY INCORPORATED, LANCET SOFTWARE DEVELOPMENT, INC., AIRLINES REPORTING CORP., MAGIC SOFTWARE ENTERPRISES LTD., MAGIC SOFTWARE ENTERPRISES, INC., TERADATA CORPORATION, EPICOR SOFTWARE CORPORATION, AND PREMIER, INC.,** **Defendants.**	**Civil Action No. 11-12220-FDS**

ORDER OF JUDGMENT

SAYLOR, J.

1. On December 26, 2012, the United States District Court for the Southern District of New York issued a final judgment in two consolidated cases, *Microsoft Corporation v. Datatern, Inc.* (11-cv-02365-KBF), and *SAP AG and SAP of America v. DataTern, Inc*., (11-cv-

02648-KBF). Among other things, that court construed certain claim limitations in United States Patent 6,101,502 ("the '502 patent"), including the claim limitation "create an interface object." That patent is the same patent at issue in the two cases presently before this Court.

2. Plaintiff DataTern, Inc., has conceded that if the court in New York correctly construed the term "create an interface object" in the '502 patent, the accused product in this case—the Business Intelligence platform of MicroStrategy, Inc.— does not infringe the '502 patent. Specifically, under that construction, the accused product here does not meet the claim limitation to "create an interface object" because it does not "generate code for at least one class and instantiate an object from that class."

3. Accordingly, and based on that concession, the motion of plaintiff DataTern, Inc., for judgment as a matter of law and the motion of defendant MicroStrategy, Inc., for summary judgment in its favor are granted in part, as further described in this order. Because the remaining defendants in both 11-cv-11970-FDS and 11-cv-12220-FDS are alleged to be merely customers of MicroStrategy, Inc., summary judgment is also granted in their favor.

4. Three of the defendants—Blazent, Inc.; Informatica, Corp.; and Airlines Reporting, Corp.—have filed counterclaims against DataTern, Inc. In those counterclaims, defendants seek declaratory judgment of non-infringement and declaratory judgment of invalidity. Defendants' requests for declaratory judgment of non-infringement are granted.

5. Final judgment of non-infringement shall enter in favor of all defendants in both cases, 11-cv-11970-FDS and 11-cv-12220-FDS.

6. In the event that the United States Court of Appeals for the Federal Circuit reverses or remands for further consideration that part of the judgment of the United States

District Court of the Southern District of New York upon which this judgment relies, upon appropriate motion this judgment shall be vacated and this Court shall conduct such further proceedings as may be appropriate.

7. To the extent that defendant MicroStrategy, Inc., sought summary judgment in its favor on other grounds, or has claimed that the '502 patent is invalid, those arguments and claims have become moot and will not be resolved at this time. Likewise, to the extent that defendants Blazent, Inc.; Informatica, Corp.; and Airlines Reporting, Corp. have sought a declaratory judgment of invalidity, those claims have become moot and will not be resolved at this time.

8. Nothing herein shall be deemed to waive any party's right to appeal from this judgment.

9. This Court will retain jurisdiction over this matter for the limited purpose of resolving defendant MicroStrategy's claims under 35 U.S.C. § 285. Defendant's motion for attorneys' fees and costs under 35 U.S.C. § 285 will remain pending, and that portion of the case will be stayed pending further order of the Court. Upon appropriate motion, the stay shall be lifted and the Court shall conduct such further proceedings as may be warranted. Any such motion to lift the stay shall be filed according to the following timetable: (a) if no appeal is taken in this matter, within 30 days after the time for filing an appeal has expired; (b) if an appeal is taken in this matter and the Court of Appeals affirms this judgment, within 30 days after affirmance; or (c) if an appeal is taken in this matter and the Court of Appeals does not affirm this judgment, at such time as the Court may direct.

So Ordered.

/s/ F. Dennis Saylor
F. Dennis Saylor IV
United States District Judge

Dated: February 7, 2013

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

DATATERN, INC., **Plaintiff,** **v.** **BLAZENT, INC.,** **Defendant.**	**Civil Action No. 11-11970-FDS**
DATATERN, INC., **Plaintiff,** **v.** **MICROSTRATEGY, INC., CARL WARREN AND COMPANY INCORPORATED, LANCET SOFTWARE DEVELOPMENT, INC., AIRLINES REPORTING CORP., MAGIC SOFTWARE ENTERPRISES LTD., MAGIC SOFTWARE ENTERPRISES, INC., TERADATA CORPORATION, EPICOR SOFTWARE CORPORATION, AND PREMIER, INC.,** **Defendants.**	**Civil Action No. 11-12220-FDS**

ORDER OF JUDGMENT

SAYLOR, J.

1. On December 26, 2012, the United States District Court for the Southern District of New York issued a final judgment in two consolidated cases, *Microsoft Corporation v. Datatern, Inc.* (11-cv-02365-KBF), and *SAP AG and SAP of America v. DataTern, Inc*., (11-cv-

02648-KBF). Among other things, that court construed certain claim limitations in United States Patent 6,101,502 ("the '502 patent"), including the claim limitation "create an interface object." That patent is the same patent at issue in the two cases presently before this Court.

2. Plaintiff DataTern, Inc., has conceded that if the court in New York correctly construed the term "create an interface object" in the '502 patent, the accused product in this case—the Business Intelligence platform of MicroStrategy, Inc.— does not infringe the '502 patent. Specifically, under that construction, the accused product here does not meet the claim limitation to "create an interface object" because it does not "generate code for at least one class and instantiate an object from that class."

3. Accordingly, and based on that concession, the motion of plaintiff DataTern, Inc., for judgment as a matter of law and the motion of defendant MicroStrategy, Inc., for summary judgment in its favor are granted in part, as further described in this order. Because the remaining defendants in both 11-cv-11970-FDS and 11-cv-12220-FDS are alleged to be merely customers of MicroStrategy, Inc., summary judgment is also granted in their favor.

4. Three of the defendants—Blazent, Inc.; Informatica, Corp.; and Airlines Reporting, Corp.—have filed counterclaims against DataTern, Inc. In those counterclaims, defendants seek declaratory judgment of non-infringement and declaratory judgment of invalidity. Defendants' requests for declaratory judgment of non-infringement are granted.

5. Final judgment of non-infringement shall enter in favor of all defendants in both cases, 11-cv-11970-FDS and 11-cv-12220-FDS.

6. In the event that the United States Court of Appeals for the Federal Circuit reverses or remands for further consideration that part of the judgment of the United States

District Court of the Southern District of New York upon which this judgment relies, upon appropriate motion this judgment shall be vacated and this Court shall conduct such further proceedings as may be appropriate.

7. To the extent that defendant MicroStrategy, Inc., sought summary judgment in its favor on other grounds, or has claimed that the '502 patent is invalid, those arguments and claims have become moot and will not be resolved at this time. Likewise, to the extent that defendants Blazent, Inc.; Informatica, Corp.; and Airlines Reporting, Corp. have sought a declaratory judgment of invalidity, those claims have become moot and will not be resolved at this time.

8. Nothing herein shall be deemed to waive any party's right to appeal from this judgment.

9. This Court will retain jurisdiction over this matter for the limited purpose of resolving defendant MicroStrategy's claims under 35 U.S.C. § 285. Defendant's motion for attorneys' fees and costs under 35 U.S.C. § 285 will remain pending, and that portion of the case will be stayed pending further order of the Court. Upon appropriate motion, the stay shall be lifted and the Court shall conduct such further proceedings as may be warranted. Any such motion to lift the stay shall be filed according to the following timetable: (a) if no appeal is taken in this matter, within 30 days after the time for filing an appeal has expired; (b) if an appeal is taken in this matter and the Court of Appeals affirms this judgment, within 30 days after affirmance; or (c) if an appeal is taken in this matter and the Court of Appeals does not affirm this judgment, at such time as the Court may direct.

So Ordered.

/s/ F. Dennis Saylor
F. Dennis Saylor IV
United States District Judge

Dated: February 7, 2013

US006101502A

United States Patent [19]

Heubner et al.

[11] Patent Number: 6,101,502

[45] Date of Patent: Aug. 8, 2000

[54] **OBJECT MODEL MAPPING AND RUNTIME ENGINE FOR EMPLOYING RELATIONAL DATABASE WITH OBJECT ORIENTED SOFTWARE**

[75] Inventors: **Robert A. Heubner**, Topsfield; **Gabriel Oancea**, Lawrence; **Robert P. Donald**, Methuen; **Jon E. Coleman**, Chelmsford, all of Mass.

[73] Assignee: **Ontos, Inc.**, Lowell, Mass.

[21] Appl. No.: **09/161,028**

[22] Filed: **Sep. 25, 1998**

Related U.S. Application Data

[60] Provisional application No. 60/069,157, Dec. 9, 1997, and provisional application No. 60/059,939, Sep. 26, 1997.

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/103**; 707/104

[58] **Field of Search** 707/100–104; 709/302

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,291,583	3/1994	Bapat	395/500
5,627,979	5/1997	Chang et al.	345/335
5,729,739	3/1998	Cantin et al.	707/104
5,752,027	5/1998	Familiar	395/614
5,809,505	9/1998	Lo et al.	707/102
5,812,996	9/1998	Rubin et al.	707/2
5,873,093	2/1999	Williamson et al.	707/103
5,878,411	3/1999	Burroughs et al.	707/4
5,937,409	8/1999	Wetherbee	707/103
5,956,725	9/1999	Burroughs et al.	707/101

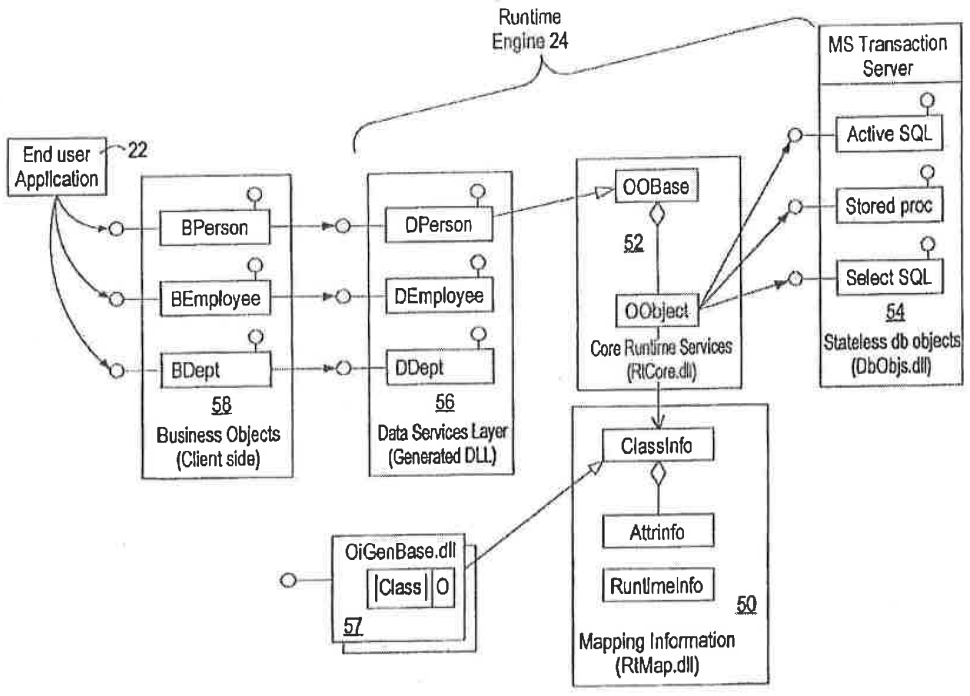
Primary Examiner—Paul V. Kulik

Attorney, Agent, or Firm—Weingarten, Schurgin, Gagnebin & Hayes LLP

[57] **ABSTRACT**

A mapping between an object model and a relational database is generated to facilitate access to the relational database. The object model can be created from database schema or database schema can be created from the object model. Further, the mapping can be automatically generated. The Database schema, object model and mapping are employed to provide interface objects that are utilized by a runtime engine to facilitate access to the relational database by object oriented software applications.

18 Claims, 6 Drawing Sheets



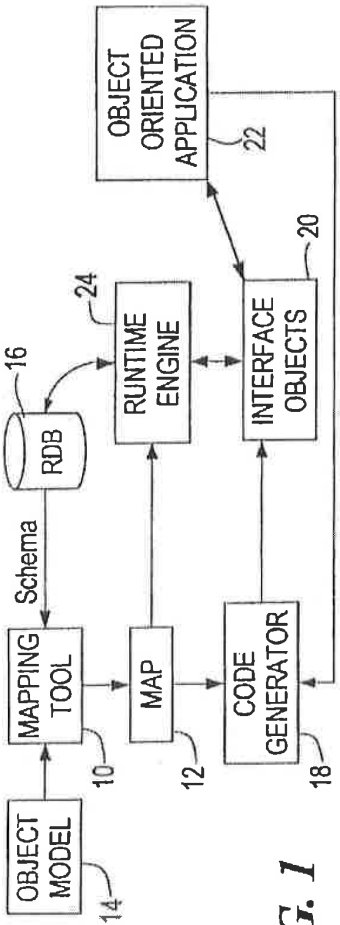

OBJECT ORIENTED APPLICATION
22
20
INTERFACE OBJECTS
24
RUNTIME ENGINE
16
RDB
Schema
MAPPING TOOL
10
MAP
12
CODE GENERATOR
18
OBJECT MODEL
14

FIG. 1

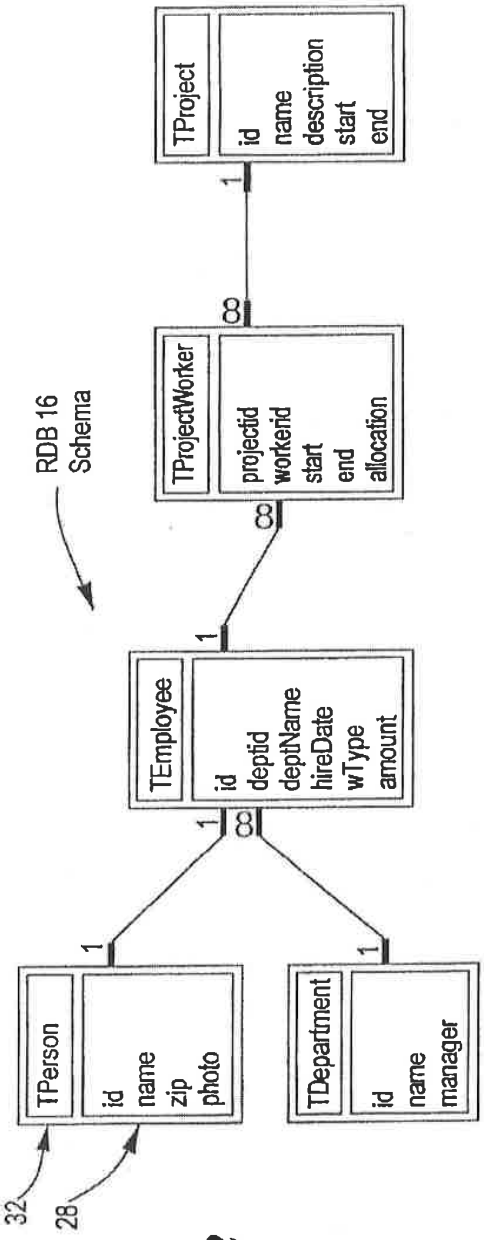

RDB 16 Schema
TProject
id
name
description
start
end
1
8
TProjectWorker
projectid
workerid
start
end
allocation
8
1
TEmployee
id
deptid
deptName
hireDate
wType
amount
1
8
1
TPerson
id
name
zip
photo
1
TDepartment
id
name
manager
32
28

FIG. 2

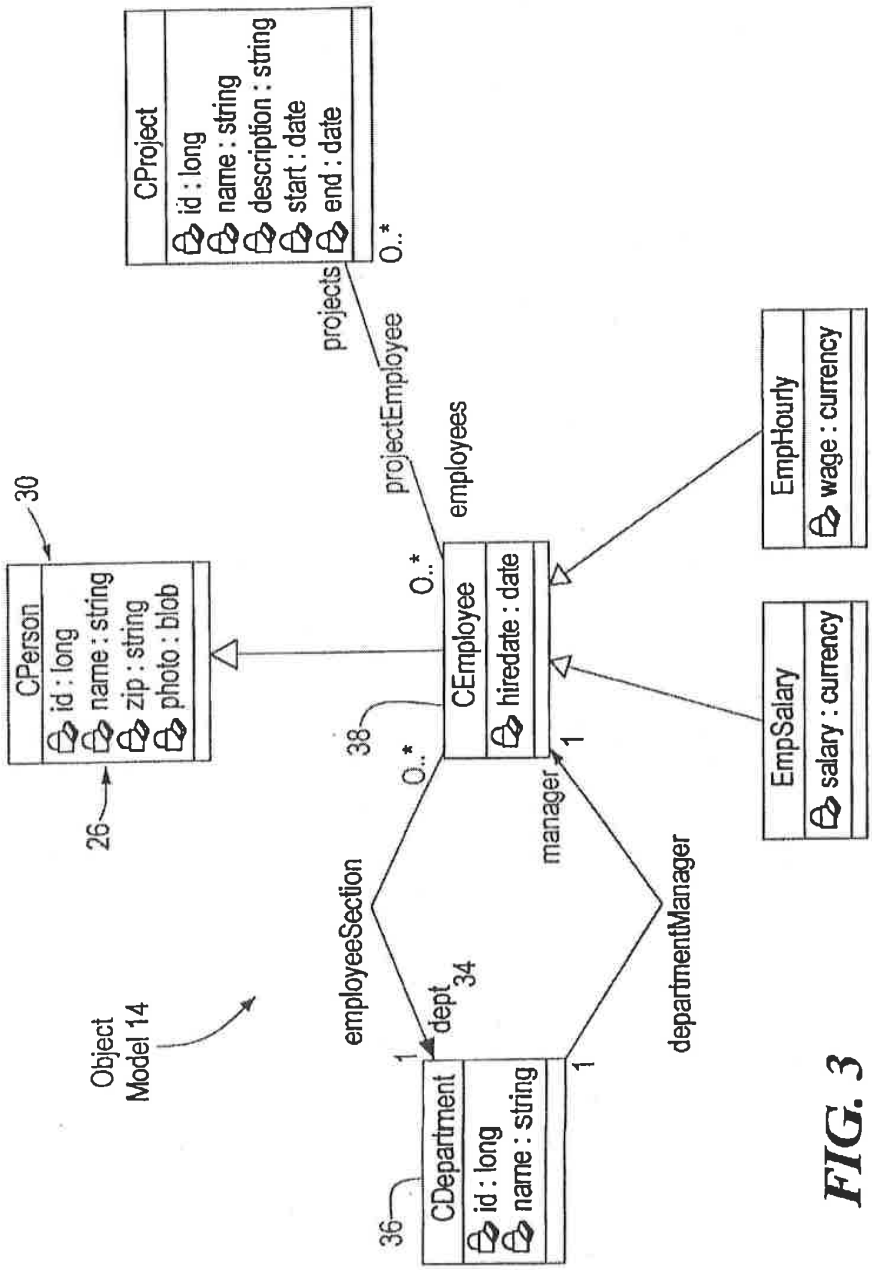

CProject
id : long
name : string
description : string
start : date
end : date
0..*
projects
projectEmployee
employees
CPerson
id : long
name : string
zip : string
photo : blob
30
26
CEmployee
hiredate : date
EmpHourly
wage : currency
EmpSalary
salary : currency
38
0..*
1
manager
employeeSection
dept
34
departmentManager
Object Model 14
CDepartment
id : long
name : string
36

FIG. 3

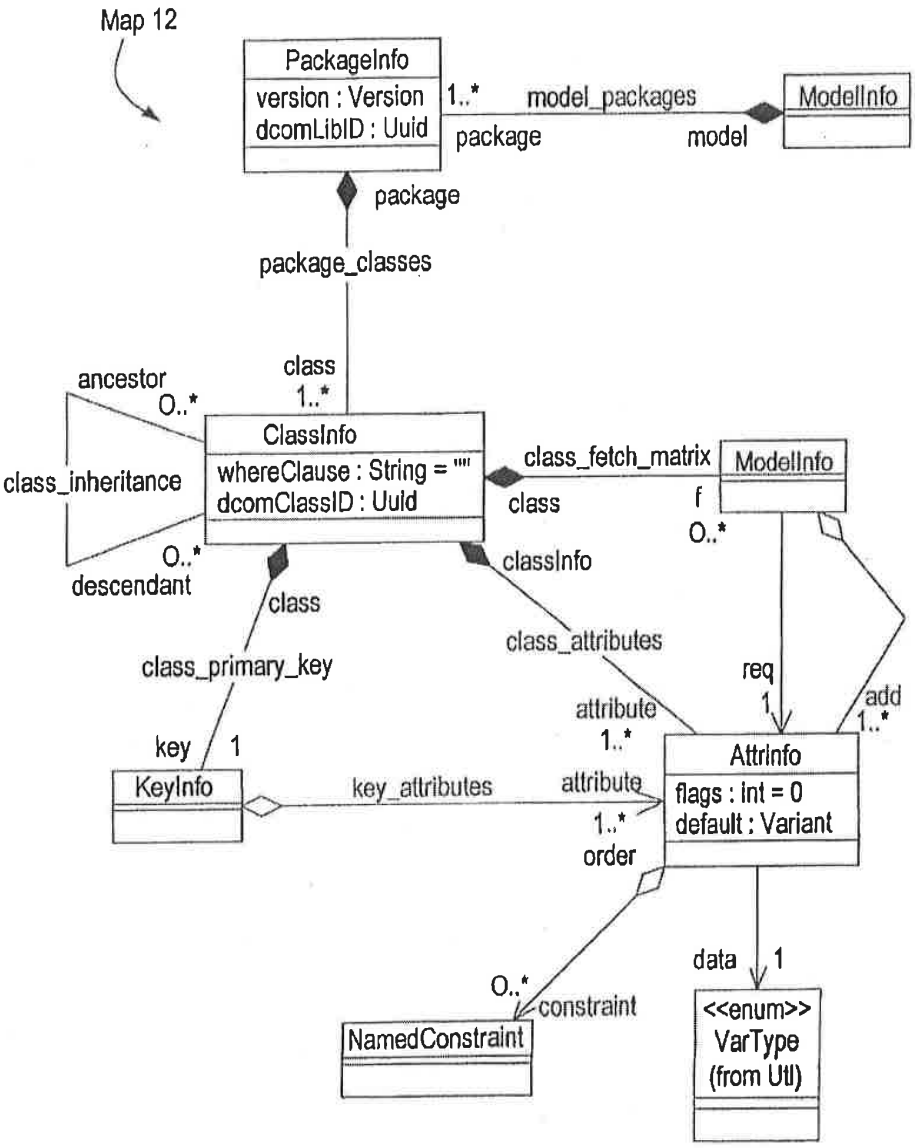

Map 12
PackageInfo
version : Version
dcomLibID : Uuid
1..*
model_packages
package
model
ModelInfo
package
package_classes
class
1..*
ancestor
0..*
ClassInfo
whereClause : String = ""
dcomClassID : Uuid
class_fetch_matrix
class
f
0..*
ModelInfo
class_inheritance
0..*
descendant
classInfo
class
class_attributes
class_primary_key
req
1
add
1..*
attribute
1..*
key
1
KeyInfo
key_attributes
attribute
AttrInfo
flags : int = 0
default : Variant
1..*
order
data
1
0..*
constraint
NamedConstraint
<<enum>>
VarType
(from Util)

FIG. 4

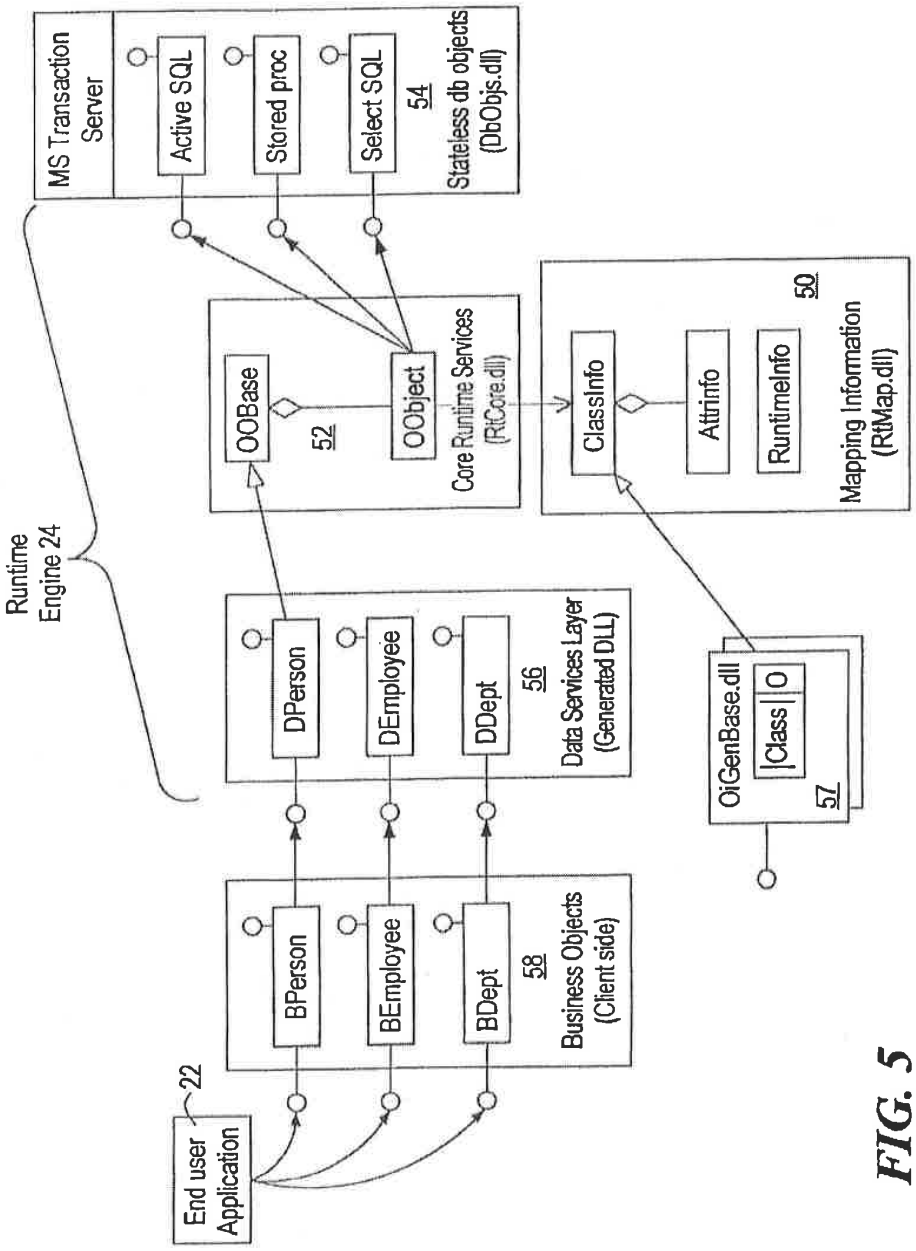

MS Transaction Server
Active SQL
Stored proc
Select SQL
54
Stateless db objects (DbObjs.dll)
OOBase
52
OObject
Core Runtime Services (RtCore.dll)
ClassInfo
AttrInfo
RuntimeInfo
50
Mapping Information (RtMap.dll)
Runtime Engine 24
DPerson
DEmployee
DDept
56
Data Services Layer (Generated DLL)
OiGenBase.dll
Class | O
57
BPerson
BEmployee
BDept
58
Business Objects (Client side)
22
End user Application

FIG. 5

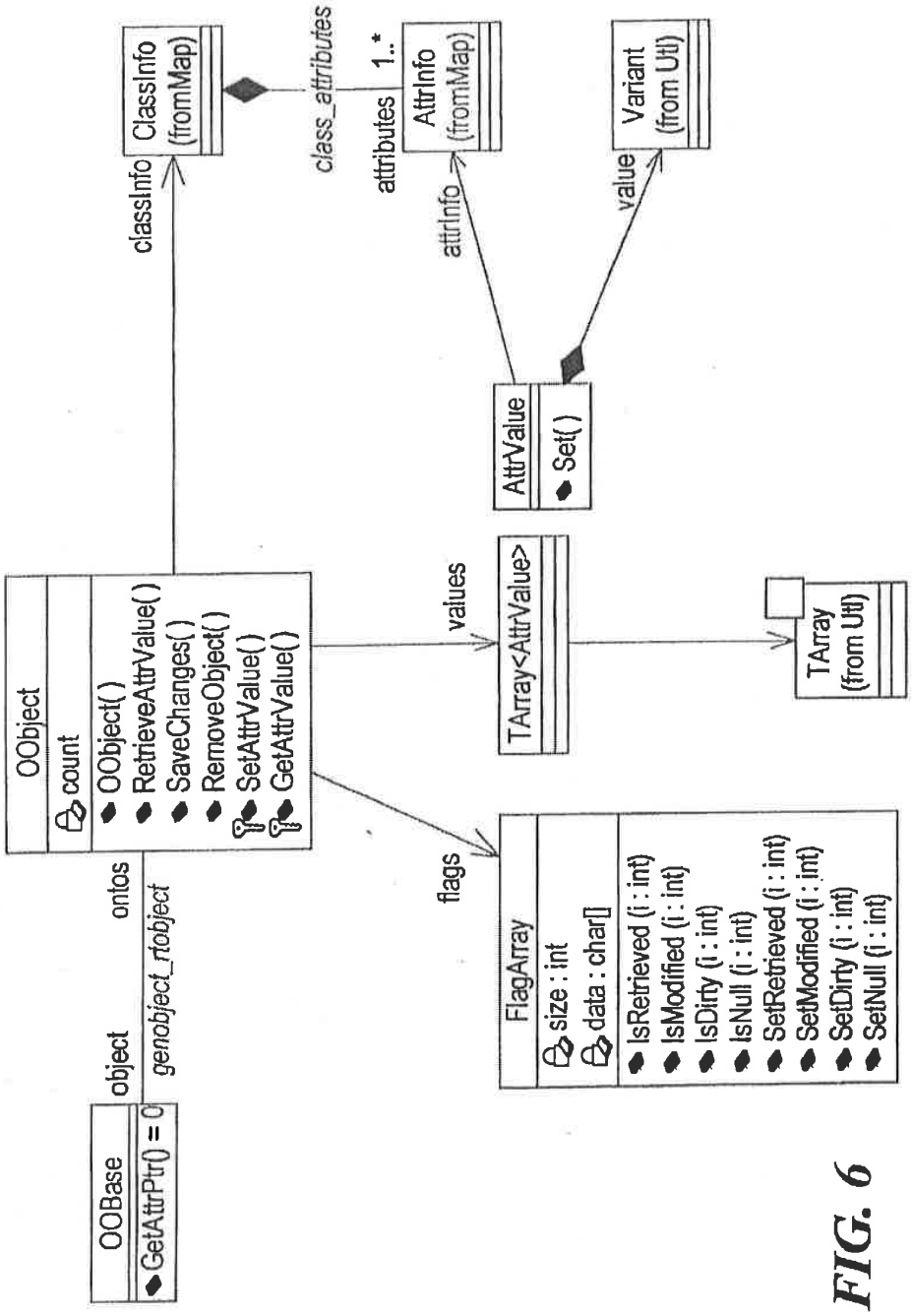

ClassInfo
(fromMap)
class_attributes
1..*
attributes
AttrInfo
(fromMap)
Variant
(from Util)
classInfo
attrInfo
value
AttrValue
Set( )
OObject
count
OObject( )
RetrieveAttrValue( )
SaveChanges( )
RemoveObject( )
SetAttrValue( )
GetAttrValue( )
values
TArray<AttrValue>
TArray
(from Util)
flags
FlagArray
size : int
data : char[]
IsRetrieved (i : int)
IsModified (i : int)
IsDirty (i : int)
IsNull (i : int)
SetRetrieved (i : int)
SetModified (i : int)
SetDirty (i : int)
SetNull (i : int)
ontos
genobject_rtobject
object
OOBase
GetAttrPtr() = 0

FIG. 6

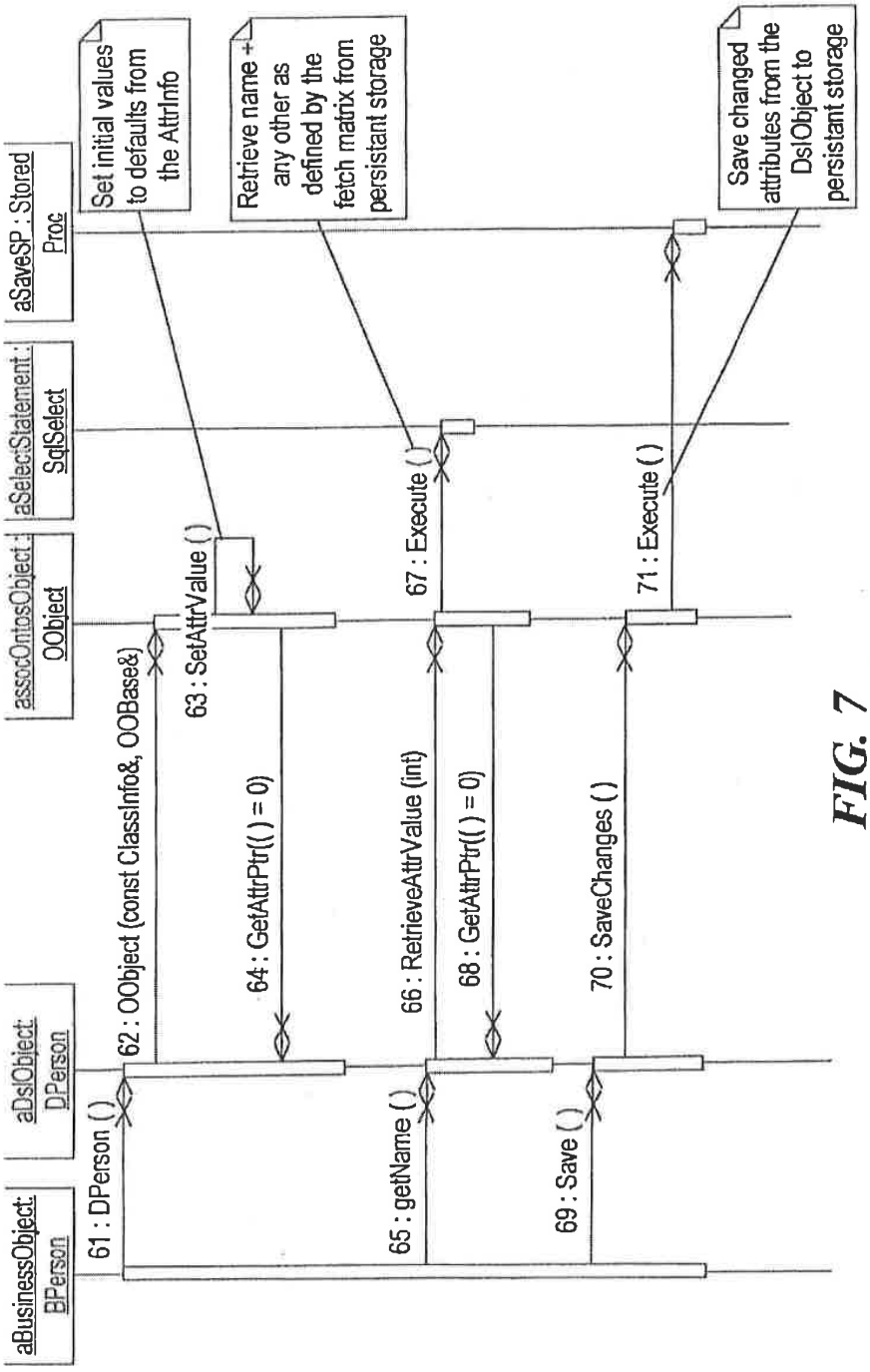

aBusinessObject: BPerson
aDslObject: DPerson
assocOntosObject: OObject
aSelectStatement: SqlSelect
aSaveSP : Stored Proc
Set initial values to defaults from the AttrInfo
Retrieve name + any other as defined by the fetch matrix from persistant storage
Save changed attributes from the DslObject to persistant storage
61 : DPerson ( )
62 : OObject (const ClassInfo&, OOBase&)
63 : SetAttrValue ( )
64 : GetAttrPtr(( ) = 0)
65 : getName ( )
66 : RetrieveAttrValue (int)
67 : Execute ( )
68 : GetAttrPtr(( ) = 0)
69 : Save ( )
70 : SaveChanges ( )
71 : Execute ( )

FIG. 7

OBJECT MODEL MAPPING AND RUNTIME ENGINE FOR EMPLOYING RELATIONAL DATABASE WITH OBJECT ORIENTED SOFTWARE

CROSS REFERENCE TO RELATED APPLICATIONS

A claim of priority is made to U.S. Provisional Patent Application Serial No. 60/069,157, entitled TIER 3 DESIGN SPECIFICATION, filed Dec. 9, 1997 and incorporated herein by reference; and U.S. Provisional Patent Application Serial No. 60/059,939, entitled DATABASE SYSTEM ARCHITECTURE, filed Sep. 26, 1997 and incorporated herein by reference.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

N/A.

BACKGROUND OF THE INVENTION

The present invention is generally related to database technology, and more particularly to interfacing object oriented software applications with relational databases.

The need for interfacing object oriented software applications with relational databases is well known. One method of interfacing an object oriented application with a relational database is to adapt the requests made by the application to the relational database. More particularly, object operations are translated into relational database queries. However, this technique is processor-intensive and sacrifices some of the advantages associated with the object oriented model. As a result, the object oriented software application is unable to function efficiently.

Another method of interfacing an object oriented application with a relational database is to translate database information into a format which is compatible with the object oriented application. Relational databases typically separate data into a plurality of tables through a process known as "normalization" to minimize duplication. A normalized relational database includes a plurality of tables, wherein each table includes at least one field and one key, and at least one field in each table is uniquely dependent upon the key that is associated with the table. These tables can be translated into objects. However, the objects can become inaccurate when changes are made to the relational database. It is known to adapt to changes in the relational database by performing further translations, but this process requires substantial effort.

BRIEF SUMMARY OF THE INVENTION

In accordance with the present invention, a mapping between an object model and a relational database and a runtime engine are employed to facilitate access to a relational database. The object model can be created from database schema or database schema can be created from the object model. Further, the mapping can be automatically generated. The database schema, object model, and mapping are employed to provide interface objects that are utilized by an object oriented software application to access the relational database.

The present invention provides transparent access to the relational database. The interface objects and runtime engine perform read and write operations on the database, including generation of SQL code. Consequently, neither programmers nor software applications need have knowledge of the database structure, the database programming interface, database security, or the database transaction model in order to obtain access to the relational database. Further, changes to the relational database do not always necessitate additional mapping.

BRIEF DESCRIPTION OF THE DRAWING

Other features and advantages of the present invention will become apparent in light of the following detailed description of the drawing, in conjunction with the drawing, of which:

FIG. **1** is a block diagram that illustrates use of the map to generate interface objects that are employed by a runtime engine and an object oriented software application to access a relational database;

FIG. **2** is a block diagram of database schema;

FIG. **3** is a block diagram of an object model;

FIG. **4** is an object diagram of a mapping;

FIG. **5** is an object diagram of the runtime engine;

FIG. **6** is an object diagram of RtCore.DLL; and

FIG. **7** is a sequence diagram that illustrates operation of the runtime engine.

DETAILED DESCRIPTION OF THE INVENTION

Referring to FIG. **1**, a mapping tool **10** is employed to generate a map **12** in which relationships between an object model **14** and schema associated with a relational database **16** are defined. A code generator **18** is employed to examine the relationships that are defined in the map **12** and a model object oriented interface associated with an object oriented software application **22** to generate interface objects **20**. The interface objects **20** are employed by the object oriented software application **22** to access the relational database **16** via a runtime engine **24**, which also uses the map **12** to drive its processing.

The object model **14** is a template that has a predetermined standardized structure. The illustrated object model includes attributes and inheritance relationships that are mapped to relational database features such as tables, rows, columns, keys, and foreign keys. Mapping the object model to the relational database schema includes mapping a class attribute to a table column, mapping a class attribute to a 1-1, 1-N, or N-N relationship, and mapping class inheritance to rows within a table or across tables.

Referring now to FIGS. **2, 3** and **4**, the mapping of a class attribute to a table column can be described generally as: Class Attribute→Table Column+Class Key+Joins. Mapping the class attribute defines where the attributes are read from and written to. In the illustrated example, the class attribute CPerson.name 26 maps to table column TPerson.name **28**. The "Class Key" is employed to relate an object instance to a row in the table. In particular, key values of the class are mapped to columns in a table that comprise the primary key. In the illustrated example, CPerson.id **30** maps to TPerson.id **32**. "Joins" defines keys between tables within a class. Since there is only one table in the mapping of Cperson.name to Tperson.name, no information is required for Joins. If Cperson includes two tables, such as Tperson and X, then mapping Cperson.y to X.y includes: Cperson.y Maps to X.y+Cperson.id Keys to Tperson.id+Tperson.id Joins to X.id.

Mapping a class attribute to a 1-1, 1-N, or N-N relationship with at least one other object can be described generally

as: Class Attribute→Class+Class Attribute→Foreign Key+Joins. When an object has associations to other objects, an attribute in the object points to one or multiple other objects. If the object points to only one object, there is a one-to-one (1-1) association between the objects. If an object points to multiple objects, there is either a one-to-many (1-N) or many-to-many (N-N) relationship between the objects. In the illustrated example, CEmployee.dept **34** maps to Cdepartment **36**, where CEmployee **38** to CDepartment **36** is a one to one relationship. "Foreign Key" represents identifying the foreign key. If CEmployee is related to CDepartment, there is a foreign key to another table. The foreign key is identified within one of the tables that comprise Cemployee and is related to the class attribute Cemployee.dept. This relationship may be inferred from foreign key information in the database schema. It is also possible that foreign key information is missing or that there are many foreign keys in CEmployee to CDept. Consequently, this step involves selecting columns that represent the foreign key. In the illustrated example, Cemployee.dept is associated with TEmployee.deptid. Once a class attribute is associated with the foreign key which resides in that class, "Joins" is defined to associated classes. In the illustrated example, TEmployee.deptid joins to TDepartment.id is defined.

Mapping class inheritance to rows within a table or across tables is performed by specifying a WHERE clause on the class which can distinguish the class from the associated parent class. This information is stored in the mapping model.

Table 1 describes how an object model can be mapped to structures in a database schema.

TABLE 1

In object model:	Can be mapped to:
A single class	All or selected columns in a table. A WHERE clause can be associated with a class to specify which rows of the table belong to the class. Multiple tables that are joined by the same primary key, or by a unique foreign key relationship. If the same data is stored in multiple tables, the duplicate columns can be handled by mapping one of the table columns for read, and all of the columns for insert and update. Multiple tables, possibly in different databases, which have similar column definitions (e.g., EastEmployees and WestEmployees tables can be merged as a single Employees class). Multiple tables that are unrelated in the database, if a logical relationship can be defined in the mapping.
Single inherited classes	A single denormalized table. Columns that contain data for all records are typically mapped to the superclass, and a WHERE clause is defined for each subclass as a discriminator for selecting which rows belong to the given subclass. Multiple tables that have the same primary keys. To ensure uniqueness of records, the primary keys of "subclass" tables may also be defined as foreign keys for the "superclass" table.
Multiple inherited classes	A single denormalized table. Different columns are mapped to each of the classes. Typically, there are multiple columns that can be used as indexes. The subclass mapping has multiple joins defined, which are used to traverse each of the inheritance relationships.

TABLE 1-continued

In object model:	Can be mapped to:
	Multiple indexed tables. The table that is mapped to the subclass has multiple keys corresponding to each of the keys in the superclass tables. To ensure uniqueness of records in the subclass table, the key on the subclass table may not be defined as a multi-column key. If all the data of the superclass tables is duplicated in the subclass table, then no join is required to instantiate an instance of the subclass. However, joins would be needed to ensure integrity of data when performing insert, remove, and update operations.

Object relationships are mapped to the database schema by defining the joins needed to access related objects or groups of objects (lists). The joins make use of foreign keys defined in the tables that are mapped to the related classes. Table 2 describes mapping of object relationships relative to the illustrated example.

TABLE 2

Object Relationship:	Mapping Example:
1-1 object relationship	Department can have only one Manager. Department class has an object attribute (Manager) that references only one instance of the Employee class. Department table has a foreign key column (ManagerID) that references only one row in the Employee table. A join is defined for the Department class based on a unique foreign key.
N-1 object relationship	Many Employees have one Department. The Employee class has an object attribute (Department) that references one instance of the Department class. The Employee table has a foreign key column (DeptID) that references one row in the Department table. A join is defined for the employee class to access the one row in the Department table that is referenced by the foreign key in the Employee table.
1-N object relationship	A Department has many Employees. The Department class has a list attribute (Employees) that references the related group of employees. Employee table has a non-unique foreign key that references the Department table. A join is defined for the Department class that selects all rows in Employee that matches the current Department's Deptid.
N-N object relationship	An Employee can have many Projects. A project can have many Employees. The Employee class has a list attribute that references a group of Projects, and the Project class has a list attribute that references a group of Employees. This mapping uses joins based on the join table that relates the Employee table and the Project table. The Employee class uses a join to select rows from the Project table that match the current instance's Employeeid. The project class uses a join to select rows from the Employee table that match the current instance's Projectid.

Table 3 describes how structures in a database schema can be mapped to structures in an object model.

TABLE 3

In a database schema:	Can be mapped to:
Rows, discriminated by WHERE clause	All attributes of a single class
A single table	All attributes of a class, assuming the other persistent attributes of the class are mapped to columns in other tables. Multiple classes (effectively, a vertical split of the table) A single-inherited classes (if at least one column is appropriate for discriminating selection of rows for subclasses) Multiple-inherited classes (if the table has multiple indexes)
Multiple tables, different columns	A single class (if the key structure exists to join row uniquely) Multiple classes (unrelated, unless key structure exists to support joins)
Multiple table, same columns	A single class that represents a logical merge of the tables. (NOTE: Primary key values must be unique between the tables.)
Multiple tables, same primary key	Single inherited classes (each table represents a class, keys used to define joins between subclass tables and superclass table. single class (with joins based on primary key)

If the same data is stored in multiple tables, the duplicate columns can be handled by mapping one of the table.columns for read, and all of the columns for insert and update.

Schema relationships are mapped directly to object relationships, either in the form of object attributes or list attributes. In general, a foreign key in the database schema is mapped to an inverse relationship between an object attribute (on the class mapped to the table holding a foreign key) and a list attribute (on the class mapped to the table referenced by a foreign key). A join table is mapped to an inverse relationship between list attributes defined on each of the classes mapped to the tables related by the join table.

Table 4 describes how relational keys are mapped to object relationships relative to the illustrated example.

TABLE 4

Schema Relationship	Corresponding Object Relationship
Unique Foreign Key	1-only-1 object relationship represented by an object attribute with a cardinality of one on the class mapped to the table that has the foreign key. This relationship can also be mapped as an embedded type.
Non-Unique Foreign Key	N-1 object relationship, represented by an object attribute on the class mapped to the table with the foreign key, and a list attribute on the class mapped to the table referenced by the foreign key.
Join Table (with no other data columns)	N-N object relationship, represented by a list attribute on each of the classes mapped to the tables related by the join. Each list attribute represents a collection of references to objects of the other type.
Join Table (with additional data columns)	A class mapped to the join table, AND a N-N object relationship, represented by a list attribute on each of the classes mapped to the tables related to the join.

Referring to FIGS. **5** and **6**, the runtime engine comprises a plurality of dynamic link libraries ("DLLs") including: RtMap.dll **50**, RtCore.dll **52**, DbObjs.dll **54**, OiGenBase.dll **57**, and a set of generated DLLs **56**. The generated DLLs **56** contain one COM interface and implementation class for each class defined by a mapping model. A mapping model binary file is generated in parallel with each DLL containing the mapping information associated with the DLL. The RtMap.dll **50** implements the classes that can load the information from the binary file at runtime and make it available to the runtime interface objects associated with DLLs **56** and to the client objects **58** of the generated COM objects through a set of predefined COM interfaces.

Classes OOBase and OObject in RtCore.dll **52** form the core of the runtime engine **24**. The OOBase is a base abstract class which is used as a base for all the generated implementation classes. The generated classes are ATL COM objects implementing one of the standard IDslObject/IDslList/IDslQlist and one or more of the client interfaces (e.g., Employee). The ATL implementation classes have state implemented as a set of attributes of the primitive types called the "front state" (or the front data set). The OOBase contains a pointer to the OObject and a public pure virtual method to access the address of each attribute in the classes descending from itself. The attributes are indexed according to the class definition for the object. The OObject class is abstracting the runtime functionality for a generic object. It contains a set of attribute info-value pairs (one per attribute, constructed when the object is initialized to form a "back state," or baseline). OObject also has a set of attribute flags (one per attribute, bitwise or of values like isModified, isRetrieved, isDirty, isNull and others). One instance of the OObject is created for every instance of the generated objects to take care of the interface to the persistent data storage through a set of DB objects that are MTS stateless, transactional objects.

FIG. **7** illustrates the sequence of actions that take place when a business object creates a Dsl object in step **61**, accesses the name property in step **65** and saves the object in step **69**. OObject is constructed when the constructor of the DPerson (the generated COM implementation class) is invoked in step **62**. The constructor passes as parameters the appropriate constant ClassInfo reference and a reference to itself. The OOBject initializes all flags and sets the attributes to the default values as defined in the AttrInfo objects associated with the ClassInfo in step **63**. The GetAttrPtr( ) function defined by the OOBase is employed to get the attribute address for each attribute in the class in order to initialize the front set of attributes on the object in step **64**.

When the getName (propget) of the generated object is called in step **65**, the generated code checks to see if the attribute was retrieved. If the attribute was retrieved then the cached value is returned. Otherwise, RetrieveAttrValue( ) of the OObject is called in step **66**, passing the id of the desired attribute (name in the example). The OObject will look at the fetch matrix for this attribute and see what other attributes should be retrieved with it in step **67** and then determines what tables and columns are involved, how they are joined and executes the appropriate SQL statements using the stateless MTS object. The GetAttrPtr( ) function defined by the OOBase is employed to get the attribute address for each attribute in the class in step **68**.

When the object is saved in step **69**, the generated code calls the OObject SaveChange( ) method in step **70**. The OObject determines what attributes have changed and, depending on the concurrency control mode in effect, makes sure the appropriate locks and transactions are set and respectively open and then executes the appropriate SQL to write the data to the persistent storage in step **71**.

Referring again to FIGS. **1–4**, the runtime engine also includes a plurality of performance enhancing features such as optimized data retrieval algorithms. An attribute retrieval

can be associated with each attribute to optimize attribute retrieval from the database. As a default case, all attributes are retrieved when any one of an object's attributes are needed. However, the attribute retrieval list for any attribute can be edited to specify different attribute retrieval behavior. For example, a request for an Employee Id may cause the Photo attribute to be dropped from the attribute retrieval list on the Id attribute if that data resides in another table and is only infrequently used. Attribute retrieval lists are a performance feature that enable optimized data access by only doing JOINS and additional SELECT statements when the data returned by those actions is needed.

Performance is also enhanced by "just in time" data retrieval. By default, whenever an attribute value is read from the database, all of the other attributes for that instance are also read. However, Data Component Developers are permitted to modify the mapping information for a Data Component to define an attribute retrieval group for each attribute of a class that determines which other attribute values are returned when the requested attribute is read from the database. This makes it possible to avoid executing JOINs or SELECTs to retrieve data that may not be needed. For example, assume that a class, CPerson, has four attributes: Id, Name, Zip, and Photo, and the Photo attribute is mapped to a column in a different table from the others. The Data Component Developer may drop Photo from the group of attributes that are retrieved when either Id, Name, or Zip are read. A query is issued to get the Name and Id of a instance of CPerson where Id=10. Based on the attribute retrieval information, the run time engine retrieves only the values for the person.id, person.name, and person.zip attributes, thus avoiding an unnecessary join to return the photo attribute value as well.

If an object does not have an attribute in memory when an attempt is made to use that attribute, the object will issue a SELECT statement to retrieve the attribute from the database. "Just-in-time" attribute population allows the object to be populated with the minimal amount of information necessary for the application while still making any remaining information available when it is needed.

Lazy reads are also employed to enhance runtime performance. When a query is defined to identify objects for retrieval from the database, the SQL SELECT statement is not issued immediately. Queries are executed only after an attempt has been made to use or modify the resulting data.

Having described the embodiments consistent with the present invention, other embodiments and variations consistent with the present invention will be apparent to those skilled in the art. Therefore, the invention should not be viewed as limited to the disclosed embodiments but rather should be viewed as limited only by the spirit and scope of the appended claims.

What is claimed is:

1. A method for interfacing an object oriented software application with a relational database, comprising the steps of:

selecting an object model;

generating a map of at least some relationships between schema in the database and the selected object model;

employing the map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application; and

utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

2. The method of claim 1 further including the step of mapping a class attribute to a table column.

3. The method of claim 1 further including the step of mapping a class attribute to a 1-1 relationship.

4. The method of claim 1 further including the step of mapping a class attribute to a 1-N relationship, where N is an integer that is greater than 1.

5. The method of claim 1 further including the step of mapping a class attribute to an N-N relationship, where N is an integer that is greater than 1.

6. The method of claim 1 further including the step of mapping class inheritance to rows within a table.

7. The method of claim 1 further including the step of mapping class inheritance across a plurality of tables.

8. The method of claim 1 further including the step of creating a plurality of said interface objects.

9. The method of claim 8 further including the step of creating at least one stateful interface object and at least one stateless interface object.

10. A computer program fixed on a computer-readable medium and adapted to operate on a computer to provide access to a relational database for an object oriented software application, comprising:

a mapping routine that generates a map of at least some relationships between schema in the database and a selected object model;

a code generator that employs said map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application; and

a runtime engine that invokes said at least one interface object to access data from the relational database.

11. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a table column.

12. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a 1-1 relationship.

13. The program of claim 10 wherein said mapping routine is operative to map a class attribute to a 1-N relationship, where N is an integer that is greater than 1.

14. The program of claim 10 wherein said mapping routine is operative to map a class attribute to an N-N relationship, where N is an integer that is greater than 1.

15. The program of claim 10 wherein said mapping routine is operative to map class inheritance to rows within a table.

16. The program of claim 10 wherein said mapping routine is operative to map class inheritance across a plurality of tables.

17. The program of claim 10 wherein said code generator is operative to create a plurality of said interface objects.

18. The program of claim 17 wherein said code generator is operative to create at least one stateful interface object and at least one stateless interface object.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,101,502
DATED : August 8, 2000
INVENTOR(S) : Robert A. Huebner et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
In the heading, "Heubner et al.", should read -- Huebner et al. --; and

Item [75], Inventors, "Robert A. Heubner", should read -- Robert A. Huebner --.

Signed and Sealed this

Twentieth Day of November, 2001

Attest:

Nicholas P. Godici

Attesting Officer

NICHOLAS P. GODICI
Acting Director of the United States Patent and Trademark Office

US006101502C1

(12) **EX PARTE REEXAMINATION CERTIFICATE** (7148th)

United States Patent
Huebner et al.

(10) **Number: US 6,101,502 C1**
(45) **Certificate Issued: Nov. 10, 2009**

(54) **OBJECT MODEL MAPPING AND RUNTIME ENGINE FOR EMPLOYING RELATIONAL DATABASE WITH OBJECT ORIENTED SOFTWARE**

(75) Inventors: **Robert A. Huebner**, Topsfield, MA (US); **Gabriel Oancea**, Lawrence, MA (US); **Robert P. Donald**, Methuen, MA (US); **Jon E. Coleman**, Chelmsford, MA (US)

(73) Assignee: **Firestar Software, Inc.**, Framingham, MA (US)

Reexamination Request:
No. 90/008,452, Jan. 25, 2007

Reexamination Certificate for:
Patent No.: **6,101,502**
Issued: **Aug. 8, 2000**
Appl. No.: **09/161,028**
Filed: **Sep. 25, 1998**

Certificate of Correction issued Nov. 20, 2001.

Related U.S. Application Data

(60) Provisional application No. 60/069,157, filed on Dec. 9, 1997, and provisional application No. 60/059,939, filed on Sep. 26, 1997.

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** ..... **707/103 R**; 707/104.1

(58) **Field of Classification Search** ..... None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,525,780	A	6/1985	Bratt et al.
4,853,843	A	8/1989	Ecklund
4,864,497	A	9/1989	Lowry et al.
4,930,071	A	5/1990	Tou et al.
4,953,080	A	8/1990	Dysart et al.
4,989,132	A	1/1991	Mellender et al.
5,008,853	A	4/1991	Bly et al.
5,010,478	A	4/1991	Deran
5,079,695	A	1/1992	Dysart et al.
5,093,914	A	3/1992	Coplien et al.
5,113,522	A	5/1992	Dinwiddie, Jr. et al.
5,125,091	A	6/1992	Staas, Jr. et al.
5,129,083	A	7/1992	Cutler et al.
5,129,084	A	7/1992	Kelly, Jr. et al.
5,136,712	A	8/1992	Perazzoli, Jr. et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CA	2147421	10/1995
WO	WO 97/07470	2/1997
WO	99/09494	2/1999

OTHER PUBLICATIONS

http://www.pcmag.com/encyclopedia_term/0,2542,t=runtime+engine&i=56079,00.asp.*

Microsoft Press. Computer Dictionary. 3rd Edition, pp. 448–449. 1997.*

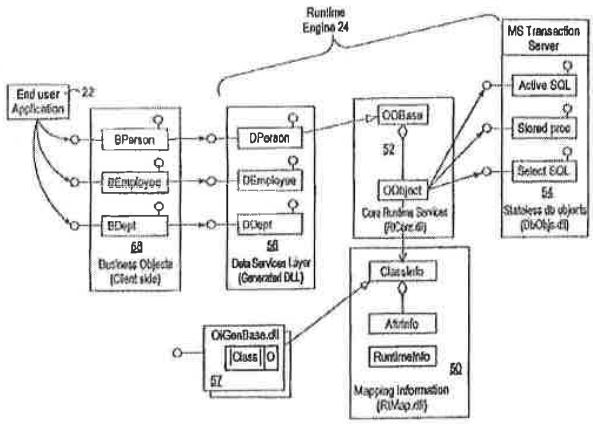
Arthur M. Keller, Richard Jensen and Shailesh Agarwal, Persistence Software: Bridging Object-Oriented Programming and Relational Databases, pp. 523–528 (1993).

(Continued)

Primary Examiner—Deandra M Hughes

(57) **ABSTRACT**

A mapping between an object model and a relational database is generated to facilitate access to the relational database. The object model can be created from database schema or database schema can be created from the object model. Further, the mapping can be automatically generated. The Database schema, object model and mapping are employed to provide interface objects that are utilized by a runtime engine to facilitate access to the relational database by object oriented software applications.



U.S. PATENT DOCUMENTS

5,142,674	A	8/1992	Barker et al.
5,157,777	A	10/1992	Lai et al.
5,161,223	A	11/1992	Abraham
5,161,225	A	11/1992	Abraham et al.
5,170,446	A	12/1992	Sullivan et al.
5,175,848	A	12/1992	Dysart et al.
5,185,885	A	2/1993	Dysart et al.
5,187,786	A	2/1993	Densmore et al.
5,187,787	A	2/1993	Skeen et al.
5,187,790	A	2/1993	East et al.
5,191,522	A	3/1993	Bosco et al.
5,193,180	A	3/1993	Hastings
5,206,951	A	4/1993	Khoyi et al.
5,212,787	A	5/1993	Baker et al.
5,227,967	A	7/1993	Bailey
5,235,701	A	8/1993	Ohler et al.
5,265,206	A	11/1993	Shackelford et al.
5,283,894	A	2/1994	Deran
5,291,583	A	3/1994	Bapat
5,291,593	A	3/1994	Abraham et al.
5,295,256	A	3/1994	Bapat
5,295,259	A	3/1994	Horne
5,315,709	A	5/1994	Alston, Jr. et al.
5,327,559	A	7/1994	Priven et al.
5,369,761	A	11/1994	Conley et al.
5,386,564	A	1/1995	Shearer et al.
5,388,264	A	2/1995	Tobias, II et al.
5,499,371	A	3/1996	Henninger et al.
5,504,885	A	4/1996	Alashqur
5,542,078	A	7/1996	Martel et al.
5,586,311	A	12/1996	Davies et al.
5,596,746	A	1/1997	Shen et al.
5,615,362	A	3/1997	Jensen et al.
5,627,979	A	5/1997	Chang et al.
5,659,723	A	8/1997	Dimitrios et al.
5,694,597	A	12/1997	Cantin et al.
5,694,598	A	12/1997	Durand et al.
5,696,961	A	12/1997	Briscoe et al.
5,701,453	A	12/1997	Maloney et al.
5,706,506	A	1/1998	Jensen et al.
5,717,924	A	2/1998	Kawai
5,727,203	A	3/1998	Hapner et al.
5,729,739	A	3/1998	Cantin et al.
5,734,887	A	3/1998	Kingberg et al.
5,737,597	A	4/1998	Blackman et al.
5,752,027	A	5/1998	Familiar
5,761,656	A	6/1998	Ben-Shachar
5,765,162	A	6/1998	Blackman et al.
5,778,375	A	7/1998	Hecht
5,809,505	A	9/1998	Lo et al.
5,812,996	A	9/1998	Rubin et al.
5,819,086	A	10/1998	Kroenke
5,829,006	A	10/1998	Parvathaneny et al.
5,850,544	A	12/1998	Parvathaneny et al.
5,857,197	A	1/1999	Mullins
5,873,093	A	2/1999	Williamson et al.
5,875,333	A	2/1999	Fish et al.
5,878,411	A	3/1999	Burroughs et al.
5,893,108	A	4/1999	Srinivasan et al.
5,897,634	A	4/1999	Attaluri et al.
5,924,100	A	7/1999	Chang et al.
5,937,402	A	8/1999	Pandit
5,937,409	A	8/1999	Wetherbee
5,956,725	A	9/1999	Burroughs et al.
5,956,730	A	9/1999	Burroughs et al.
5,977,967	A	11/1999	Berner et al.
6,047,284	A	4/2000	Owens et al.
6,061,515	A	5/2000	Chang et al.
6,076,090	A	6/2000	Burroughs et al.
6,078,926	A	6/2000	Jensen et al.
6,085,198	A	7/2000	Skinner et al.
6,108,664	A	8/2000	Nori et al.
6,122,627	A	9/2000	Carey et al.
6,134,540	A	10/2000	Carey et al.
6,163,776	A	12/2000	Periwal
6,173,290	B1	1/2001	Goldberg
6,175,837	B1	1/2001	Sharma et al.
6,223,227	B1	4/2001	Williamson et al.
6,226,637	B1	5/2001	Carey et al.
6,240,422	B1	5/2001	Atkins et al.
6,374,252	B1	4/2002	Althoff et al.
6,385,618	B1	5/2002	Ng et al.
6,466,992	B2	10/2002	Williamson et al.
6,704,744	B1	3/2004	Williamson et al.
6,952,706	B2	10/2005	Williamson et al.
7,127,474	B2	10/2006	Williamson et al.

OTHER PUBLICATIONS

InfoBase Software Specification, pp. 1–121 (1990).

InfoBase User Manual, pp. 1–50 (1989).

Weiyi Mang, Clement Yu, Wong Kim, Gaoming Wang, Tracy Pham and Son Dao, Construction of a Relational Front–end for Object–Oriented Database Systems, pp. 476–483 (1993).

Michael R. Blaha, William J. Premerlani, and James E. Rumbaugh, Relational Database Design Using an Object–Oriented Methodology, pp. 414–427 (1988).

C. Souza dos Santos and E. Theroude, Persistent Java, pp. 1–22 (1996).

Red Hat, Inc. & Red Hat Middleware, LLC, Defendants Red Hat, Inc. and Red Hat Middleware, LLC'S Invalidity Contentions, litigation document, Aug. 6, 2007, pp. 1–41, United States District Court For The Eastern District of Texas Marshall Division.

Red Hat, Inc. & Red Hat Middleware, LLC, Defendants Red Hat, Inc. and Red Hat Middleware, LLC'S Supplemented Invalidity Contentions, litigation document, Feb. 7, 2008, pp. 1–266 (including the Exhibits), United States District Court For The Eastern District of Texas Marshall Division.

Civil Action No. 2:06–CV–258 (TJW), entitled as "Joint Claim Construction and Prehearing Statement," pp. 1–7.

Civil Action No. 2:06–CV–258 (TJW), entitled as "Firestar Software, Inc's Claim Construction Brief Pursuant to P.R.4–5(a)" pp. 1–35 (excluding exhibits).

Civil Action No. 2:06–CV–258 (TJW), entitled as "Red Hat's Claim Construction Brief" pp. 1–34.

Civil Action No. 2:06–CV–258 (TJW), entitled as "Firestar Software, Inc.'s Reply Brief Pursuant to P.R.4–5(c)" pp. 1–17.

Civil Action No. 2:06–CV–258 (TJW), entitled as "Red Hat's Sur–Reply Brief Regaring Claim Construction" pp. 1–7.

Rotterdam, Ernest, OORL: A Declarative and Object Oriented Database Language (Aug. 8, 1995).

Presentations of the OOPSLA '95 Workshop: Objects and Relational Databases, (Oct. 16, 1995).

Abiteboul, Serge, O2 Java ODMG Binding Presentation, May 23, 1997 ("JRB Presentation").

Application Development—Oracle Plans to Pave Object Path With Sedona Project, Infoworld Publishing Company (Aug. 13, 1995).

Barsalou, Thierry, et al., Complex Objects for Relational Databases, Computer Aided Design, vol. 22, No. 8 ("Complex Objects").

Barsalou, Thierry, et al., Knowledge–Directed Mediation Between Application Objects and Base Data, Data Knowledge Base Integration: Proceedings of the Working Conference ("Knowledge–Directed Mediation").
Barsalou, thierry, et al., Updating Relational Databases Through Object–Based Views, AM SIGMOD International Conference on Management of Data, vol. 20, No. 2 ("Updating Relational Databases").
Campbell, The Object Oriented Design and Implementation of a Rlational Database Management System.
Core Functionality for Object to Relations Maps, Oracle Corporation (1996).
Data Access Builder ("DAX Builder").
Data Access Builder Implementation Overview ("DAX Implementation Overview").
Data Access Builder Window ("DAX Builder Window").
Documentation relating to Pocket (1996–1997), produced by Sun under separate cover.
Enterprise Object Framework Developer's Guide: NeXTStep Developer's Library.
Enterprise Objects Framework Developer's Guide: OpenStep Developer's Library, NeXT Computer, Inc. (1996) ("OpenStep Developer's Library").
Fishman, D.H., et al., "IRIS: An Object–Oriented Database Management System," ACM Transactions on Office Information Systems, vol. 5, No. 1 ("Fishman '87").
Fitsilis, P., Producting Relational Database Schemata from an Object Oriented Design, IEEE (1994) ("Producing Relational Database Schemata").
Gardarin, Georges, et al. Managing Complex Objects in an Extensible Relational DBMS, Proceedings of the Fifteenth International Conference on Very Large Data Bases pp. 55–56 ("Gardarin 1989").
InfoBase for Smalltalk –80 Release 1.0 External reference Specification, Rev. 2.2 ("Infobase ERS").
InfoBase User's Manual, ParcPlace Systems, Inc. Rev. 4 ("InfoBase Manual").
Introduction to TOPLink Version 2.1 ("TOPLink for SmallTalk v. 2.1 Manual").
Java Blend/Pocket ("Java Blend/Pocket").
Java Blend source code (1996–1997).
John Petrie, "Object Integration Server" Presentations of OOPSLA '95 Workshop Objects and Relational Databases ("ONTOS OIS Presentation").
John Petrie, "ONTOS Object Integration Server (ONTOS OIS): Integrating Objects with Relational Databases Technical Overview" Proceedings of the OOPSLA '95 Workshop Objects and Relational Databases ("ONTOS OIS Technical Overview").
Joss Object Services Persistence Service Specification, OMG TC No. 93.11.3 ("JOSS").
Kelley, W. et al., "Schema Architecture of the UniSQL/M Multidatabase System," Modern Database Systems: The Object Model, Interoperability, and Beyond ("UniSQL/M Schema Architecture").
Kleindienst, Jan, et al., "Lessons Learned from Implementing the CORBA Persistent Object Service" OOPSLA '96 ("Kleindienst").
Kleissner, Charly "Enterprise Objects Framework: A Second Generation Object–Relational Enabler" SIGMOD, Jun. 1995 ("Kleissner EOF").
Klug, A., et al., "Multiple View Support within the ANSI/SPARC Framework," Proceedings of the Third International Conference on Very Large Databases ("Klug").
Kyriakakis, Ted, Subtleware Post (Feb. 8, 1994 ("Subtleware Post").
Law, Kincho et al., Managing Design Objects in a Sharable Relational Framework (1990) ("Managing Design Objects").
Law, Kincho, Management of Complex Structural Engineering Objects in a Relational Framework, Engineering with Computers, Issue 6 (1990) ("Complex Structural Engineering Objects").
Lee, B. et al., "Efficiency instantiating view–objects from remote relational databases," The International Journal on Very Large Data Vases, vol. 3, No. 3, pp. 289–323 ("Lee VLDB 1994").
Lee, B. et al., Outer joins and filters for instantiating objects from relational databases through views, IEEE Transactions on Knowledge and Data Engineering (Feb. 1994) vol. 6, No. 1 p. 108–19 ("Lee IEEE 1994").
Manola, Frank Integrating Object Oriented Applications and Middleware with Relational Databases, GTE, Laboratories, Inc. ("Manola").
"Marketing Brief on Java Blend" (Mar. 20, 1997).
NeXTStep Enterprise Objects Framework Developer's Guide: NeXTStep Developer's Library, Release 3 (1994).
Novera EPIC DB Blend ("Novera").
O2 Java Relational Binding, at http://web.arquive.org/1997022231614/http://www.o2tech.fr/ (Feb. 26, 1996) ("O2 JRB Website").
"Object/Relational Integration: ONTOS Object Integration Server" Advertisement (May 1996).
ONTOS Object Integration Server for Relational Databases 1.0: Schema Mapper User's Guide, (ONTOS, Inc. 1989–1994) (Nov., 1994), Proceedings of the OOPSLA '95 Workshop: Objects and Relational Databases, (Oct. 16, 1995).
ONTOS OIS for Relational Databases 1.0: Schema Mapper User's Guide ("ONTOS SMUG").
Oracle7 Server SQL Reference Manual: Elements of Oracle7 SQL ("Oracle Elements").
Oracle7 Server SQL Reference Manual: Commands ("Oracle Commands").
Oracle Floats Plan, Computerworld via First! By Individual, Inc. (Feb. 19, 1995).
Oracle's public deminstrations of the product in 1996.
Persistence Demonstration for SIGMOD '93 ("Persistence Video").
Persistence Demonstration for SIGMOD '93 Video, and Persistence User Manual, Version 1.2 (Persistence Software 1992) (Mar. 1993).
Persistence User Manual—Version 1.2 ("Persistence Manual").
Proceedings of the OOPSLA '95 Workshop: Objects and Relational Databases, (Oct. 16, 1995).
Pröll, Elisabeth, Design of a Meta Data Base for Mapping Complex Objects in a Relational Data Base ("COMan Thesis").
Reinwald, Storing and Using Objects in a Relational Database, IBM Systems Journal, vol. 35, No. 2 ("Reinwald").
Rowe, L., "A Shared Object Hierarchy," (Revised) ("Rowe '87").
Sedona Build 092 Oracle Corp. ("Sedona Code").
Sedona Product Demonstration ("Sedona Product Demo").
Sedona v. 1.00 Installation and Release Notes, (Oracle Corporation 1996).

Souza dos Santos, C., Persistent Java, First International Workshop on Persistence and Java ("Persistent Java").
Srinivasan, V., Object Persistence in Object Oriented Applications, IBM Systems Journal, vol. 36, No. 1, 66–87 ("Srinivasan Persistence").
Subtleware for C++/Oracle: User Guide (1993–1995) ("Subtleware Users Guide").
Subtleware, Demonstration Guide (1995) ("Subtleware Demonstration Guide").
Sujanski, Walter, Structural Semantics and Complex Objects for the Coupling of Relational Databases and Knowledge–Based Systems, (1990) ("Coupling Relational Databases and Knowledge–Based Systems").
Thelen, F. et al., "POET SQL Object Factory: Technical Overview" ("POET").
TOPLink for Smalltalk Version 2.0 distributed code ("Toplink for V 2.0 Code").
TOPLink for SmallTalk Version 4.0 Manual (TOPLink for SmallTalk V. 4.0).
TOPLink Java Version 1.0 User's Manual ("TOPLink for Java V 1.0 Manual").
TOPLink Smalltalk Version 2.0 User's Manual ("TOPLink for SmallTalk V. 2.0 Manual").
Verona Applications Requirements & Applications Framework Strategy ("Verona Framework").
Verona Major Features Design Document: Applications Technology Object–Oriented Applications ("Verona Major Features").
Visual Works Object Reference, Rev. 1.1 ("VisualWorks Object Reference").
Visual Works User's Guide, Rev. 2.0 ("VisualWorks User's Guide").
Wiederhold, Gio, et al., "Integrating Data into Objects Using Structural Knowledge," Third International Symposium on Command and Control Research and Technology, National Defense University ("Integrating Data into Objects").

* cited by examiner

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

NO AMENDMENTS HAVE BEEN MADE TO THE PATENT

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims 1–18 is confirmed.

New claims 19–44 are added and determined to be patentable.

19. The method of claim 1 further comprising reading the map with the runtime engine to at least one of, read data from and write data to the relational database.

20. The method of claim 19 wherein the reading the map comprises loading information from the map at runtime.

21. The method of claim 1 further comprising creating an instance of a runtime engine class by the runtime engine to abstract relational database access functionality from the object oriented software application, wherein the relational database access functionality comprises reading and writing data.

22. The method of claim 21 wherein the creating comprises creating one instance of the runtime engine class by the runtime engine for every interface object instance.

23. The method of claim 19 further comprising generating at least one mapping model file containing at least some mapping information from the map, wherein the generating at least one mapping model file occurs prior to utilizing the runtime engine.

24. The method of claim 23 wherein reading the map further comprises loading information from one or more mapping model files by the runtime engine.

25. The method of claim 1 further comprising executing a query by the runtime engine to one of, retrieve or store particular data after detecting a need to one of, use or modify the particular data by the software application.

26. The method of claim 19 further comprising executing a query by the runtime engine to one of, retrieve or store particular data after detecting a need to one of, use or modify the particular data by the software application, wherein executing the query occurs after reading the map.

27. The method of claim 19 wherein utilizing the runtime engine comprises invoking at least one interface object after reading the map with the runtime engine.

28. The computer program of claim 10 wherein said runtime engine comprises at least one shared library.

29. The computer program of claim 28 wherein said at least one shared library includes executable loading code for the loading mapping information from the map.

30. The computer program of claim 28 wherein said at least one shared library includes executable abstracting code for abstracting relational database access functionality from the object oriented software application, wherein the relational database access functionality includes reading and writing data.

31. The computer program of claim 28 wherein said at least one shared library includes at least one dynamic link library (DLL).

32. The method of claim 1 wherein said utilizing is further defined as the object oriented software application invoking at least one interface object to request data from the relational database corresponding to at least one attribute of at least one object corresponding to a class associated with the object oriented software application.

33. The method of claim 32 wherein said utilizing is further defined as at least one interface object invoking the runtime engine to obtain data requested by the object oriented software application.

34. The method of claim 33 wherein said utilizing is further defined as at least one interface object calling a method on a runtime engine object.

35. The method of claim 33 wherein said utilizing is further defined as the runtime engine accessing the database to obtain the data requested by the object oriented software application.

36. The method of claim 35 wherein said utilizing is further defined as the runtime engine reading the map to determine at least one table and at least one column that needs to be accessed to obtain the data requested by the object oriented software application.

37. The method of claim 35 wherein said utilizing is further defined as the runtime engine invoking at least one database object to execute a database query.

38. The method of claim 33 wherein said utilizing is further defined as the runtime engine invoking at least one interface object to deliver the data requested by the object oriented software application.

39. The method of claim 1 further defined as creating the object model corresponding to the object oriented software application.

40. The method of claim 39 further defined as creating schema in the database based on the object model.

41. The method of claim 1 further defined as creating schema in the database.

42. The method of claim 41 further defined as creating the object model corresponding to the object oriented software application based on schema in the database.

43. The computer program of claim 10 wherein the runtime engine includes a runtime engine class for defining at least one method to abstract relational database access functionality from the object oriented software application, and wherein the relational database access functionality comprises reading and writing data.

44. The computer program of claim 43 wherein an instance of the runtime engine class is created for each interface object instance.

* * * * *

United States Court of Appeals
for the Federal Circuit
DATATERN, INC. v BLAZENT, INC., 2013-1251, -1252

CERTIFICATE OF SERVICE

I, Robyn Cocho, being duly sworn according to law and being over the age of 18, upon my oath depose and say that:

Counsel Press was retained by McCARTER & ENGLISH, LLP, Attorneys for Plaintiff-Appellant to print this document. I am an employee of Counsel Press.

On **May 28, 2013**, Counsel for Appellant has authorized me to electronically file the foregoing **Brief of Plaintiff-Appellant** with the Clerk of Court using the CM/ECF System, which will serve via e-mail notice of such filing to any of the following counsel registered as CM/ECF users:

Richard G. Frenkel
Melissa A. Kopacz
Latham & Watkins LLP
140 Scott Drive
Menlo Park, CA 94025
650-463-3080
rick.frenkel@lw.com
melissa.kopacz@lw.com
Counsel for Blazent, Inc.

Gregory A. Madera
Adam J. Kessel
Sivananda Kumjula Reddy
Fish & Richardson, P.C.
One Marina Park Drive
Boston, MA 02210
617-521-7809
madera@fr.com
kessel@fr.com
reddy@fr.com
Counsel for Microstrategy, Inc.

Heather B. Repicky
Nutter McClennen & Fish LLP
Firm: 617-439-2000
Seaport West, 155 Seaport Boulevard
Boston, MA 02110
617-439-2192
hrepicky@nutter.com
Counsel for Informatica Corp.

Benjamin K. Thompson
Fish & Richardson, P.C.
1180 Peachtree Street
21st Floor
Atlanta, GA 30309
404-724-2844
bkt@fr.com
Counsel for Microstrategy, Inc.

Derek B. Domian
Goulston & Storrs
400 Atlantic Avenue
Boston, MA 02110-3334
617-574-6568
ddomian@goulstonstorrs.com
Counsel for Lancet Software Development, Inc.

Steven M. Coyle
Cantor Colburn LLP
55 Griffin Road South
Bloomfield, CT 06002
860-286-2929
scoyle@cantorcolburn.com
Counsel for Magic Software Entertainment, Inc.

Stephen R. Buckingham
Lowenstein Sandler, P.C.
65 Livingston Avenue
Roseland, NJ 07068
973-597-2500
sbuckingham@lowenstein.com
Counsel for Epicor Software Corp.

Brian P. Voke
Campbell, Campbell, Edwards & Conroy, P.C.
One Constitution Plaza
Charlestown, MA 02129
617-241-3000
bvoke@campbell-trial-lawyers.com
Counsel for Carl Warren & Co., Inc.

Dominic E. Massa, Esq.
Wilmer Cutler Pickering Hale and Dorr LLP
60 State Street
Boston, MA 02109
617-526-6000
dominic.massa@wilmerhale.com
Counsel for Airlines Reporting Corp.

Paper copies will also be mailed to the above counsel at the time paper copies are sent to the Court.

Upon acceptance by the Court of the e-filed document, six paper copies will be filed with the Court, via Federal Express, within the time provided in the Court's rules.

May 28, 2013

/s/Robyn Cocho
Robyn Cocho
Counsel Press

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS AND TYPE STYLE REQUIREMENTS

1. This brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a)(7)(B).

__X__ The brief contains 8,173 words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii),or

____ The brief uses a monospaced typeface and contains _____ lines of text, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii).

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6).

__X__ The brief has been prepared in a proportionally spaced typeface using MS Word 2002 in a 14 point Times New Roman font or

____ The brief has been prepared in a monospaced typeface using MS Word 2002 in a ___ characters per inch_____ font.

May 28, 2013
Date

(s) /s/ Erik P. Belt
Erik P. Belt
Counsel for Appellant